Contributions to indexing and retrieval using Formal Concept Analysis

Victor Codocedo-Henriquez

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Contributions à l'indexation et à la recherche d'information avec l'analyse formelle de concepts

THÈSE

présentée et soutenue publiquement le 4 de Septembre de 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Víctor Codocedo-Henríquez

Composition du jury

Président :	Dr. François CHAROY - LORIA, Université de Lorraine (Nancy, France)
Rapporteurs :	Dr. Patrick GALLINARI - LIP6, Université Pierre et Marie Curie (Paris, France) Dr. Céline ROBARDET - LIRIS, INSA de Lyon (Lyon, France)
Examinateurs :	Dr. Claudio CARPINETO - Fondazione Ugo Bordini (Rome, Italy) Dr. Peter EKLUND - IT University of Copenhagen (Copenhagen, Denmark) Dr. Marianne HUCHARD - LIRMM, Université de Montpellier (Montpellier, France)
Encadrants :	Dr. Hernán ASTUDILLO - Universidad Federico Santa María (Valparaíso, Chile) Dr. Amedeo NAPOLI - Directeur de recherche CNRS (Nancy, France)

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Contents

Introduction 1

Chapter 1
Theoretical Background & State-of-the-art

1.1 Introduction . . . 9
1.2 Formal Concept Analysis . . . 10
1.2.1 Implications, Definitions and Association Rules . . . 11
1.2.2 Conceptual Stability . . . 13
1.2.3 Many-valued Contexts . . . 14
1.2.4 Conceptual Scaling . . . 15
1.2.5 Calculating a concept lattice . . . 16
1.2.6 Pattern structures . . . 16
1.3 Information Retrieval . . . 18
1.4 A survey on Formal Concept Analysis based Information Retrieval approaches . . . 20
1.4.1 Pre-FCA history - A lattice to model the description and document spaces 22
1.4.2 FCA meets IR . . . 24
1.4.3 Enriching the description space through external knowledge sources . . . 26
1.4.4 Relevance Feedback and Automatic Retrieval . . . 27
1.4.5 Applications and Systems . . . 29
1.5 Conclusions . . . 31

Chapter 2
Contributions on Retrieval

2.1 Prologue . . . 33
2.2 Introduction . . . 34
2.3 Background . . . 35
2.3.1 Information content as a semantic relation measure . . . 35
2.3.2 The principles of Concept Lattice-based Ranking . . . 36
2.4 CLAIRE - Concept Lattices for Information Retrieval . . . 40

2.4.1 Motivation for a new approach for Information Retrieval based on Formal Concept Analysis . . . 40
2.4.2 The principles of CLAIRE . . . 41
2.4.3 The implementation of CLAIRE as a Knowledge Discovery in Databases process . . . 42
2.4.4 Step 1 - Document Classification . . . 42
2.4.5 Step 2 - Concept Lattice Navigation . . . 43
2.4.6 Step 3 - Formal Concept Ranking . . . 46
2.5 Experimental Evaluation . . . 46
2.5.1 Experimental setting . . . 47
2.5.2 Evaluation measures . . . 47
2.5.3 Results . . . 50
2.5.4 Query analysis . . . 51
2.6 Conclusions . . . 53

Chapter 3
Contributions on Indexing

3.1 Introduction . . . 55
3.2 Background . . . 56
3.2.1 The vector space model for retrieval (VSM) . . . 56
3.2.2 Interval Pattern Structures . . . 58
3.2.3 Relational Concept Analysis (RCA) . . . 59
3.3 CLAIRE and the vector space model . . . 61
3.3.1 Querying . . . 62
3.3.2 Retrieving documents with ip-CLAIRE . . . 64
3.3.3 Experimental results . . . 64
3.3.4 Discussion on the capabilities of ip-CLAIRE . . . 65
3.4 A model for heterogeneous indexing . . . 66
3.4.1 Inspiring problem - Latent Semantic Indexing . . . 67
3.4.2 Adapting RCA for pattern structures . . . 69
3.4.3 Discussion on the heterogeneous pattern structures model . . . 74
3.5 Conclusions . . . 76

Chapter 4
Beyond indexing: Biclustering and its applications

4.1 Introduction . . . 79
4.2 Background . . . 80

4.2.1 Biclustering . . . 80
4.2.2 The links between FCA and biclustering . . . 82
4.2.3 Triadic Concept Analysis. . . . 82
4.3 Biclustering using FCA . . . 83
4.3.1 Biclustering using partitions . . . 83
4.3.2 Formalizations . . . 83
4.3.3 Partition Space and Partition Pattern Structures . . . 85
4.3.4 Experiments . . . 88
4.3.5 Discussion on Biclustering and FCA . . . 89
4.4 FCA and Biclustering: Two additional models . . . 90
4.4.1 Scaled Partition Pattern Structures . . . 91
4.4.2 Interval Pattern Structure Approach . . . 92
4.4.3 Triadic Concept Analysis Approach . . . 93
4.4.4 Experiments . . . 94
4.4.5 Discussion . . . 96
4.5 Applications . . . 96
4.5.1 Recommender Systems . . . 96
4.5.2 Functional Dependencies . . . 101
4.6 Conclusions . . . 107

Conclusions and perspectives

Bibliography **111**

List of Tables

1 Document corpus - Example . . . 3
2 Measuring documents distance - Example . . . 5

1.1 Formal Context - Example 1 . . . 11
1.2 Concept Stability - Example 1 . . . 14
1.3 Concept Stability - Example 2 . . . 14
1.4 Concept Stability - Example 3 . . . 15
1.5 Many-valued context - Example 1 . . . 15
1.6 Scaling a many-valued context - Example . . . 16
1.7 Formal Context - Example 2 . . . 18
1.8 A document-term formal context . . . 24

2.1 Experiments - Dataset Characteristics . . . 47
2.2 Precision and Recall - Examples . . . 48
2.3 Precision and Recall - Results . . . 48
2.4 Results - Comparison among approaches . . . 51
2.5 Results - Query generators for query 6 . . . 51
2.6 Results - Ranked concepts for query 6 . . . 53
2.7 Results - Query generators for query 8 . . . 53
2.8 Results - Ranked results for query 8 . . . 54

3.1 Many-valued formal context of term frequencies in each document. . . . 58
3.2 Relational context family (RCF) - Example . . . 60
3.3 Context $\mathcal{K}_1$ after relational scaling using existential quantifier. We have removed the relational attribute $\exists \texttt{aw} : \texttt{CO}$ usually assigned to every object . . . 61
3.4 CISI dataset - Results for 35 queries . . . 65
3.5 Document representations in two latent variables . . . 68
3.6 Graphical representation of documents using LSA . . . 68
3.7 Hybrid formal context - Scaling a pattern structure . . . 71
3.8 Heterogeneous pattern concepts . . . 73
3.9 Representational context of a heterogeneous pattern structure . . . 75

4.1 Constant value biclusters . . . 80
4.2 Bicluster with similar values ($\theta = 1$) - Example . . . 81
4.3 Constant column bicluster - Example . . . 81
4.4 Biclusters with similar values from pp-lattice . . . 87
4.5 Results - Comparison between CC and pp-lattice bicluster algorithm . . . 89
4.6 A numerical data table - Example . . . 91

4.7 Formal context scaled from Table 4.6 91
4.8 Triadic context derived from Table 4.6 using $\simeq_1$ 94
4.9 Results - Comparison of models for biclustering 95
4.10 A many-valued context for recommendation 98
4.11 Biclusters for recommendation 100
4.12 Functional dependencies - Datasets characteristics 106
4.13 Functional Dependencies - Results 106
4.14 Functional Dependencies - Tolerance blocks statistics 107

List of Figures

1 A general model for an IR system 2

1.1 Concept Lattice - Example 1 11
1.2 The powerset $\wp(\mathtt{M})$ represented as a lattice 19
1.3 Mapping $\delta : \mathtt{G} \rightarrow \mathtt{D}$ 20
1.4 Set pattern structure derived from Table 1.7 21
1.5 Mooers' model for IR 23
1.6 Retrieval paradigm examples 29

2.1 A term taxonomy - Example 36
2.2 Document term concept lattice 37
2.3 Hierarchical exploration strategy - Example 39
2.4 Neighbourhood expansion strategy - Example 40
2.5 3-step KDD-like document retrieval process. 42
2.6 Classification-based Reasoning algorithm 44
2.7 Cousin Concepts - Example 45
2.8 Results - 11-point interpolated precision for each dataset 52

3.1 A semi-lattice representation of intervals 59
3.2 Concept lattice of formal context $\mathcal{K}_2$ in Table 3.2b 60
3.3 Interval pattern concept lattice derived from Table 3.1 63
3.4 Interpolated precision in 11 points of recall 65
3.5 Annotated document vector space using heterogeneous pattern structures 76

4.1 Partition pattern concept lattice - Example 86
4.2 Results - AddIntent Iterations per prune vs Execution time 89
4.3 Partition pattern concept lattice and scaled concept lattice 92

Introduction

Information Retrieval (IR) is one of the most popular and widely used computer-assisted processes which users from all levels (novices and experts) access every day for a myriad of different tasks. For example, a single *smartphone* may have multiple IR systems for searching phone numbers, browsing images, music, videos, the Web, maps, books, etc. We stress the fact that a smartphone is a personal and portable device which is usually constantly connected to the network and thus, these IR systems are just those built-in the device. Through smartphones, people can instantly access large collections of documents such as Wikipedia[1], or giant search engines such as Google or Yahoo which direct them to any part of the Web.

Nevertheless, IR systems can also be much simpler, small and even non-computer-assisted. The introductory book of Information Retrieval written by Manning [93] states this fact as follows:

"Just getting a credit card out of your wallet so that you can type in the card number is a form of information retrieval."

Indeed, we can understand IR not just as computer systems, but also as a cognitive paradigm [74] which models the overlapping space between a user's information needs and a search space.

What do we understand in this thesis as an IR system? Let us land the definition of an IR system as follows: *"An IR system is a piece of software that is able to retrieve documents using an index of some sort, given a user request in the form of a query"*. Therefore, we will divide our understanding of an IR system in two main processes, namely *indexing* and *retrieval*. Figure 1 shows a general model for an IR system, represented by the inner rectangle, and both processes represented by the external rectangles[2]. The process of indexing refers to the transformation of the document collection to its description space using a representation function. The description space is traditionally called "Index". The process of retrieval refers to taking the user query to the index using a representation function. This last function may or may not be the same one used to transform documents. Once the query is positioned into the index, a retrieval function compares the query representation with document representations in order to evaluate those that may be relevant for the user. These documents become the result which is retrieved to the user.

Usually, we could consider that the process of indexing is prior to the process of retrieval. This is true in most cases, since in order to retrieve documents we need an index. However, there are some approaches that build the index at retrieval time, thus inverting the order between the processes [25, 82]. This is specially useful for composite IR systems, i.e. IR systems that do not work directly with the document corpora, but with the results from other IR systems.

In this thesis we present our contributions in the field of formal concept analysis [63] and information retrieval [93]. This document is divided in four chapters. Chapter 1 introduces the

[1] `http://www.wikipedia.org/`

[2] Taken from `http://www.slideshare.net/mounialalmas/aggregation-for-searching-complex-information-spaces`

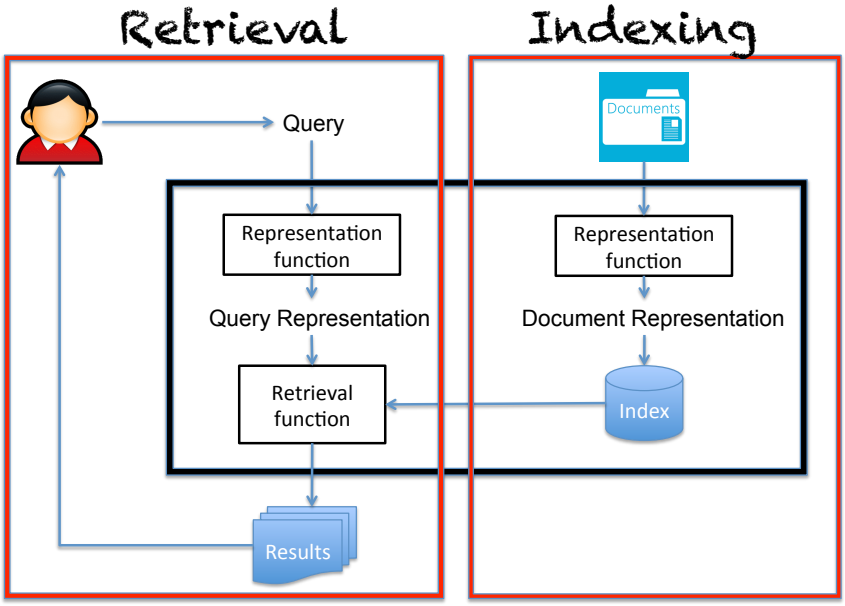


Figure 1: A general model for an IR system

main concepts and presents the state-of-the-art of IR systems supported over FCA techniques. Chapter 2 presents our contributions in the process of retrieval, while Chapter 3 presents contributions in the process of indexing. As explained in the first chapter, the model of indexing using FCA is old and well established. In the beginning of our research, we used this model to test and discover its drawbacks and benefits. Thus, our research starts with retrieval, instead of indexing. Chapter 4 presents our contributions on data mining derived from the indexing techniques we introduced in the previous chapter, particularly we present a model for biclustering. In the following, we provide an extended introduction of our work.

Chapter 1

IR systems started almost with informatics itself, covering an old but complex problem: How to search for documents in a library? However, the initial relation between document search and information management was not clear in the beginning. A rather surprising quote of Robert A. Fairthorne stated [54]:

> *"At first sight Library Classification and Information Theory appear to have little in common (...). Library Classification is concerned, or so I am often told, with knowledge. This I doubt. Information Theory is concerned (...) with the physical problems arising in the handling and reproduction of certain marked events - signals- (...)."*

Indeed, information theorists were focused in the study of signals, information flow and noise. It was in this paradigm that they undertook the document retrieval problem. Let us quote Fairthorne another time on this matter [54]:

> *"Library Classification may or may not be concerned with human knowledge, truth and falsehood, Hegelian absolutes, and the like; it is most certainly concerned with Printed Matter. That is, with large, heavy objects with marks on."*

This notion of *objects with marks on* was the norm on library classification at the time. For example, the Dewey Decimal System for library classification proposed in 1876[3] was composed of a taxonomy of topics tied to a numeric coding which were used to *mark* books within a library providing them with an order to ease their search.

The first IR models exploited this *objects with marks* view in a cleverly designed mathematical framework that could be supported by a computer in order to automate the search for documents. Later on, this model would be generalized for performance comparison purposes [105] introducing the basic notions of what two decades later would be called the framework of Formal Concept Analysis (FCA) [63]. We will refer to this model as the Boolean model of retrieval or Boolean IR. The description of these connections and how IR and FCA have evolved through the years is given in Chapter 1. This chapter also presents the theoretical framework on which the contributions of this thesis are supported.

Chapter 2

The initial quote of Fairthorne is surprising in more than one manner. For example, he says *"Library Classification is concerned, or so I am often told, with knowledge. This I doubt."* What he probably meant is that the organization of documents has little to do with the knowledge within those documents, and more with the relations among them. Indeed, if we consider documents as objects with marks, we are exploiting the fact that a single mark is applied to more than one object and thus, we can create a structure that allows us to navigate the document space, disregarding what those marks mean or what those objects contain. Let us illustrate the benefits and drawbacks of this model with the following example.

D1	きれいな花を敷きつめてあげる
D2	花が咲いているよ色とりどりの花が
D3	未来も過去も意味をもとない
D4	遠すぎた闇を越え飛び立つ
D5	この気持ちは枯れない花のように揺らめいて
D6	時は奏でて想いはあふれる
D7	そんな飛べない無邪気な天使にも

Table 1: Document corpus of Japanese documents

Table 1 shows 8 documents the content of which are excerpts from Japanese song lyrics. For any non-Japanese speaker, every character may be taken just as a symbol or a mark. Thus, we can express our retrieval system in terms of FCA where each document is an object and each character is an attribute and thus, retrieving documents can be achieved by the derivation operator applied to the symbol, e.g. $\{花\}' = \{D1, D2, D5\}$.

This example makes clear the fact that we do not even have to understand the language of a document corpus in order to be able to retrieve documents from it. This is the main benefit of this retrieval model. By ignoring the knowledge within a document collection, the retrieval model becomes *content independent*. However, this is also the reason behind its main limitation. Let us consider a new document to include in the collection.

D8 Las florés de mi jardín han de ser mis enfermeras

[3] `https://archive.org/details/classificationan00dewerich`

Document D8 presents the excerpt of a song lyrics in Spanish. It is clear that this song has absolutely no intersections with songs in Table 1 and thus, it cannot be included in the same index for retrieval purposes. In order to achieve this, it is necessary to stop ignoring what the marks mean. In fact, it is necessary to consider *knowledge* as an important resource for retrieval. For this example, knowing that the symbol 花 is an ideogram for "flower" and that "flores" is the plural for "flor" which is "flower" in Spanish would allow us to insert song D8 within the index created from Japanese lyrics. However, in this scenario documents are not related by "marks", but by "meaning" or "knowledge units", i.e. elements that map to a cognitive concept such as the concept of "flower". Let us say that the element *flower.s.1* corresponds to such concept, then $\{flower.s.1\}' = \{D1, D2, D5, D8\}$. This example may seem far-fetched, but in fact this is a very common scenario and an active research subject called Cross-Language IR [111].

The inclusion of external knowledge sources is one of the main advantages that FCA brought to the table of IR and it has been explored in different approaches [21, 120] which we review in Section 1.4.3, Chapter 1. In this thesis, we present a novel characterization of this knowledge inclusion within an IR system based on FCA through a knowledge discovery in databases process [16]. Our characterization (named CLAIRE: Concept Lattices for Information REtrieval), relies on the notion of semantic similarity to relate documents by information content[4], instead by just marks. This model, its application and experimental evaluation is fully described in Chapter 2. Its main contributions are:

- The formalization of an IR system as a knowledge discovery process
- The inclusion of semantic resources to evaluate document similarity
- The use of a case-based reasoning technique to modify a query based on semantic similarity based on the notion of *cousin-concepts*
- The proposition of a novel technique of navigation and ranking for document retrieval using a concept lattice
- The evaluation of the approach versus state-of-the-art retrieval techniques

Chapter 3

Although the Boolean IR model is still in use in several applications, it has been considered from long ago as *too limited for modern information retrieval needs [93].* Indeed, the IR community would quickly move away from the simple "object with marks" paradigm and would start to consider documents as complex objects with multiple description dimensions. These dimensions comprise different representation levels, for example document authorship, language, date of creation and abstract representations. Regarding the last one, the vector space model (VSM) [129] is one of the most popular ones.

In VSM, documents are represented as coordinates in an arbitrary vectorial space. This representation is derived from the same document content using a given representation function. For example, in the case of text documents the representation function generates a vector in which each dimension is represented by the occurrence frequency of a given term inside the document.

[4] A measure related to information entropy [93].

Query	Representation	Euclidean distance
IR	$\langle 1, 0 \rangle$	0.21
smartphone	$\langle 0, 1 \rangle$	1.20

Table 2: Euclidean distance from query representations to the document $\langle 0.85, 0.15 \rangle$

Let us illustrate this by analysing the text of this introduction up to this point. Terms information and retrieval[5] have been mentioned five times (together), while its acronym IR has been mentioned 17 times. The term smartphone has been mentioned 3 times. Of course, this is no surprise since the subject of this introduction is to present a thesis work related to information retrieval while smartphone was mentioned as a context of the narration. Instead, in a report about smartphones we would expect that this term is mentioned with much higher frequency than information retrieval. However, in the Boolean IR model, this document would be as equally relevant to the search for the term smartphone or the search for terms information retrieval. In the Boolean retrieval model a document is either *relevant* or *irrelevant* (thus, the Boolean part of the model)[6]. Instead, in the VSM we can distinguish among degrees of relevance using a distance measure. For example, let us simply consider terms IR and smartphone to describe this document. Since IR has been mentioned 17 times and smartphone only 3, we will say that our document has a length of 20 occurrences. Thus, we can represent it as the vector $\langle 0.85, 0.15 \rangle$, where 0.85 is the frequency of IR (17 over 20) and 0.15 is the frequency of smartphone (3 over 20). Given a query with the term IR, we can represent it in the same space of two dimensions, this is the frequency for the term IR in the query (1 over 1) and the frequency of the term smartphone (0 over 1) $\langle 1, 0 \rangle$. Similarly, for the query with the term smartphone its representation is given by $\langle 0, 1 \rangle$. Table 2 shows the Euclidean distance from the query representations to the document representation in the vectorial space of two dimensions. We can see how the query for IR is much closer to the document than the query smartphone, rendering it more relevant.

Using this notion, the VSM ranks documents w.r.t. the query using a comparison function. In this case we have used the Euclidean distance, but cosine similarity and other measures have also been extensively used [6]. The VSM cannot be directly supported by FCA given the Boolean nature of the latter. In fact, FCA is a natural implementation of the Boolean retrieval model. In Chapter 3 we present and adaptation of the CLAIRE system for retrieval using formal concept analysis supporting the dynamics of the VSM. We achieve this by implementing the interval pattern structures framework [79] and a notion of upper bounding the Euclidean distance among a group of documents to provide a natural raking using the concept lattice structure. We denominate this approach ip-CLAIRE (interval pattern-CLAIRE).

In the VSM, the vector representation of documents makes us go back to the content independent retrieval system. That is, we ignore the content of documents and the meaning of the vectors. In the second part of Chapter 3, we present an extension of ip-CLAIRE considering a heterogeneous representation of document descriptions supported over a novel instance of the pattern structures extension of FCA, namely heterogeneous pattern structures (HPS). Through this model we can index documents simultaneously in a vector space as well as in an external knowledge source such as a dictionary. The main contributions of Chapter 3 are:

[5]We will use this font to distinguish terms from actual "concepts". For example, "information retrieval" refers to the research field, while information retrieval refers to a piece of text.

[6]Document *relevance* will be discussed throughout this thesis in different sections and under different circumstances. For now, consider that a relevant document is one that satisfies the user information needs. A retrieval system consists on evaluating which documents may or may not be relevant given a user query.

- The presentation of a novel approach based on FCA supporting the dynamics of the vector space model for retrieval which we call ip-CLAIRE (interval pattern concept lattice for information retrieval)
- The experimental evaluation of ip-CLAIRE and comparison with current retrieval techniques
- The introduction of the *heterogeneous pattern structures*, an instance of the pattern structures framework supporting object descriptions in heterogeneous spaces
- The presentation of the *heterogeneous pattern structures* (HPS) as a technique to support *relational concept analysis* [71] over pattern structures
- The introduction of a technique to index documents in a vectorial space as well as in an external knowledge source in the form of a dictionary

Chapter 4

The frontier between indexing and mining (data mining) is fuzzy and fundamentally defined by the application they support. This fact is even clearer with the use of HPS for document indexing.

HPS are able to characterize documents with complex descriptions in different and distinct spaces. Particularly, we are interested in characterizing *clusters* of documents with *clusters* of terms, and a vector of interval of values which represent how those terms and documents relate between the clusters. Such patterns represent a *kind* of bicluster, i.e. a cluster relation through some value restriction [91].

Biclustering have been extensively used in bioinformatics for data mining in gene and protein datasets [61], information retrieval [44] and more recently, for functional dependency mining [9].

While HPS resemble biclusters, they are conceptually different. In HPS there is a representation function from the space of documents to the space of values and a second representation function from the space of documents to the space of terms. In a bicluster, documents, terms and values are bound together into a single function which is later restricted to characterize the bicluster.

In Chapter 4 we move forward indexing into data mining, more specifically we introduce a FCA-based model for biclustering using the partition pattern structures framework. Through our approach we are able to enumerate all possible biclusters from a numerical data table of objects and attributes. Moreover, by introducing a tolerance relation we show how we are able to mine biclusters based on similarity rather than equality. In order to better characterize links between biclustering and FCA, we present two additional models for bicluster mining based on interval pattern structures and triadic concept analysis [88]. Finally, we present two applications of biclustering for recommender systems and functional dependency mining. The model and applications presented in Chapter 4 wrap coherently the lessons learnt throughout Chapters 2 and 3. The main contributions of Chapter 4 are:

- The introduction of a novel bicluster mining model using partition pattern structures
- The introduction of a tolerance relation to support biclusters based on similarity
- An evaluation of the proposed model comparing with a state-of-the-art biclustering technique

- The presentation of different strategies for recommendation purposes using biclusters and a concept lattice
- The introduction of a method for functional dependency mining using partition pattern structures

Publications

The content of this thesis is based on the following publications.

- Chapter 1 is based on:
  - ▷ "Formal Concept Analysis and Information Retrieval - A survey" at the International Conference on Formal Concept Analysis. 2015 [39].
- Chapter 2 is based on:
  - ▷ "A contribution to semantic indexing and retrieval based on FCA - An application to song datasets" at Concept Lattices and their Applications. 2012 [35].
  - ▷ "A semantic approach to Concept Lattice-based Information Retrieval" at Annals of Mathematics and Artificial Intelligence. 2014 [36].
- Chapter 3 is based on:
  - ▷ "Using pattern structures to support information retrieval with Formal Concept Analysis" at 2nd Workshop "What can FCA do for Artificial Intelligence?". 2013 [34].
  - ▷ "A proposition for combining pattern structures and relational concept analysis" at International Conference on Formal Concept Analysis. 2014 [37].
- Chapter 4 is based on:
  - ▷ "Lattice-based biclustering using Partition Pattern Structures" at European Conference on Artificial Intelligence. 2014 [38].
  - ▷ "Three Related FCA Methods for Mining Biclusters of Similar Values" at Concept Lattices and their Applications. 2014 [76].

Chapter 1

Theoretical Background & State-of-the-art

Contents

1.1 Introduction 9
1.2 Formal Concept Analysis 10
1.2.1 Implications, Definitions and Association Rules 11
1.2.2 Conceptual Stability 13
1.2.3 Many-valued Contexts 14
1.2.4 Conceptual Scaling 15
1.2.5 Calculating a concept lattice 16
1.2.6 Pattern structures 16
1.3 Information Retrieval 18
1.4 A survey on Formal Concept Analysis based Information Retrieval approaches 20
1.4.1 Pre-FCA history - A lattice to model the description and document spaces 22
1.4.2 FCA meets IR 24
1.4.3 Enriching the description space through external knowledge sources . . 26
1.4.4 Relevance Feedback and Automatic Retrieval 27
1.4.5 Applications and Systems 29
1.5 Conclusions 31

1.1 Introduction

This chapter introduces the main theoretical framework over which the contributions of this thesis are supported, namely formal concept analysis (FCA) and information retrieval (IR), and in a second part it presents a survey on the relation of both of these techniques and how it has evolved through the years.

Our goal in this chapter is to contextualize the reader and provide him with a frame of reference allowing him to better understand the rationale behind our work. We approach this goal in a rather didactic manner since sadly, among the many things FCA and IR have in common, *notation is not one of them.*

1.2 Formal Concept Analysis

In the following, we provide a description of formal concept analysis using the notations of [63]. Data is encoded in a formal context $\mathcal{K} = (\mathtt{G}, \mathtt{M}, \mathtt{I})$, i.e. a binary table where $\mathtt{G}$ is a set of objects, $\mathtt{M}$ a set of attributes, and $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M}$ an incidence relation indicating by $\mathtt{gIm}$ that the object $\mathtt{g}$ has the attribute $\mathtt{m}$ (an equivalent notation for this relation is $(\mathtt{g}, \mathtt{m}) \in \mathtt{I}$). For $\mathtt{A} \subseteq \mathtt{G}$ and $\mathtt{B} \subseteq \mathtt{M}$, two derivation operators $(\cdot)'$ are defined as follows:

$$' : \wp(\mathtt{G}) \longrightarrow \wp(\mathtt{M}), \text{ with } \mathtt{A}' = \{\mathtt{m} \in \mathtt{M} \mid \forall \mathtt{g} \in \mathtt{A}, \mathtt{gIm}\} \tag{1.1}$$

$$' : \wp(\mathtt{M}) \longrightarrow \wp(\mathtt{G}), \text{ with } \mathtt{B}' = \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{B}, \mathtt{gIm}\} \tag{1.2}$$

Where $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ respectively denote the powersets of $\mathtt{G}$ and $\mathtt{M}$. The two derivation operators $(\cdot)'$ form a Galois connection[7] between $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ [63]. For a set of objects $\mathtt{A}$, $\mathtt{A}'$ is the set attributes which are common to all objects in $\mathtt{A}$. Analogously, for a set of attributes $\mathtt{B}$, $\mathtt{B}'$ is the set of objects having all attributes in $\mathtt{B}$. A *formal concept* is defined as a pair $(\mathtt{A}, \mathtt{B})$ where $\mathtt{A} \subseteq \mathtt{G}$, $\mathtt{B} \subseteq \mathtt{M}$, $\mathtt{A}' = \mathtt{B}$ and $\mathtt{B}' = \mathtt{A}$, $\mathtt{A}$ being the *extent* and $\mathtt{B}$ the *intent* of the formal concept (in this case $\mathtt{A}'' = \mathtt{A}$ and $\mathtt{B}'' = \mathtt{B}$).

Within the set of all formal concepts $\mathfrak{B}(\mathtt{G}, \mathtt{M}, \mathtt{I})$, the order $\leq_{\mathcal{K}}$ between two formal concepts is defined as follows:

$$(\mathtt{A}_1, \mathtt{B}_1) \leq_{\mathcal{K}} (\mathtt{A}_2, \mathtt{B}_2) \iff \mathtt{A}_1 \subseteq \mathtt{A}_2 \text{ (dually } \mathtt{B}_2 \subseteq \mathtt{B}_1) \tag{1.3}$$

Where $(\mathtt{A}_1, \mathtt{B}_1)$ is called the sub-concept of $(\mathtt{A}_2, \mathtt{B}_2)$ and inversely, $(\mathtt{A}_2, \mathtt{B}_2)$ is called the super-concept of $(\mathtt{A}_1, \mathtt{B}_1)$. The *concept lattice* derived from the formal context $\mathcal{K}$ is denoted by $\underline{\mathfrak{B}}(\mathtt{G}, \mathtt{M}, \mathtt{I})$.

For an object $\mathtt{g} \in \mathtt{G}$, the *object intent* is defined Equation 1.4. Correspondingly, for $m \in M$, the *attribute extent* is defined in Equation 1.5.

$$\mathtt{g}' = \{\mathtt{m} \in \mathtt{M} \,|\, \mathtt{gIm}\} \tag{1.4}$$

$$\mathtt{m}' = \{\mathtt{g} \in \mathtt{G} \,|\, \mathtt{gIm}\} \tag{1.5}$$

For a given object $\mathtt{g}$, the *object concept* is defined in Equation 1.6 (where $\mathtt{g}''$ stands for $(\mathtt{g}')'$). Dually, for a given attribute $\mathtt{m}$, the *attribute concept* is defined in Equation 1.7. Intuitively, the *object concept* is the smallest-extent concept in the lattice which includes the object. The *attribute concept* is the smallest-intent concept which contains the attribute.

$$\gamma(\mathtt{g}) = (\mathtt{g}'', \mathtt{g}') \tag{1.6}$$

$$\mu(\mathtt{m}) = (\mathtt{m}', \mathtt{m}'') \tag{1.7}$$

Finally, given a formal concept $\mathtt{C} = (\mathtt{A}, \mathtt{B})$, its support is defined by the ratio between the cardinality of its extent and the cardinality of the set of objects $\mathtt{G}$ as defined in Equation 1.8.

$$\sigma(\mathtt{C}) = \frac{|\mathtt{B}|}{|\mathtt{G}|} \tag{1.8}$$

	m_1	m_2	m_3	m_4	m_5
g_1	×	×	×		
g_2	×	×	×	×	
g_3		×	×	×	×
g_4	×			×	×

Table 1.1: Formal Context Example

Example 1. Table 1.1 contains an example formal context $\mathcal{K} = (\mathtt{G}, \mathtt{M}, \mathtt{I})$ with four objects ($\mathtt{G} = \{g_1, g_2, g_3, g_4\}$) and five attributes ($\mathtt{M} = \{m_1, m_2, m_3, m_4, m_5\}$). Each cross in the table represents an object-attribute relation and thus the incidence relation set has a cardinality of fourteen ($|\mathtt{I}| = 14$). Consider the set $\mathtt{B}_1 = \{m_3, m_4\}$, then $\mathtt{B}_1' = \mathtt{A} = \{g_2, g_3\}$. We can see that $(\mathtt{A}, \mathtt{B}_1)$ is not a formal concept since $\mathtt{A}' = \{m_2, m_3, m_4\} \neq \mathtt{B}_1$. Instead, with $\mathtt{B} = \{m_2, m_3, m_4\}$ the couple $(\mathtt{A}, \mathtt{B})$ is a proper formal concept.

Figure 1.1 illustrates the Hasse Diagram[8] of the concept lattice derived from the formal context in Table 1.1. The concept lattice is presented in **reduced notation**, i.e. only object and attribute concepts are labelled. The diagram can be read as follows: "Each node represents a formal concept. For a given concept C, all the objects with an object concept "below" C (i.e. which are sub-concepts of C) constitute its extent. Similarly, all the attributes with an attribute concept "over" C (i.e. which are super-concepts of C) constitute its intent". For example, the formal concept marked as C in the diagram has as extent the set $\{g_2, g_3\}$ and as intent the set $\{m_2, m_3, m_4\}$.

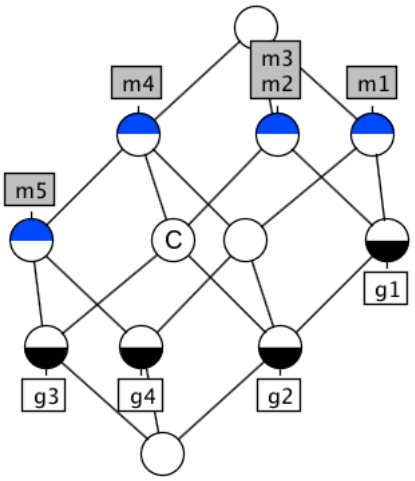


Figure 1.1: Concept Lattice derived from the formal context example in Table 1.1

.

1.2.1 Implications, Definitions and Association Rules

An implication can be established between set of attributes $\mathtt{B}_1, \mathtt{B}_2 \subseteq \mathtt{M}$ with $\mathtt{B}_1 \cap \mathtt{B}_2 = \emptyset$ as follows:

[7] A Galois connection is based on a dual adjunction between partially ordered sets.
[8] Figure obtained with ConExp software `http://conexp.sourceforge.net/`.

$$\mathtt{B}_1 \rightarrow \mathtt{B}_2 \iff \mathtt{B}_2 \subseteq \mathtt{B}_1'' \tag{1.9}$$

Equation 1.9 is read as "$\mathtt{B}_1$ *implies* $\mathtt{B}_2$" iff all the objects containing attributes in $\mathtt{B}_1$ also contain the attributes in $\mathtt{B}_2$. We also say that $\mathtt{B}_1$ is a *premise* of $\mathtt{B}_2$.

Example 2. Consider the attribute sets $\mathtt{B}_1 = \{\mathtt{m}_3, \mathtt{m}_4\}$ and $\mathtt{B}_2 = \{\mathtt{m}_2\}$ in Table 1.1, then:

$$\begin{aligned} \mathtt{B}_1' &= \{\mathtt{g}_2, \mathtt{g}_3\} \\ \mathtt{B}_1'' &= \{\mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4\} \\ \mathtt{B}_2 \subseteq \mathtt{B}_1'' &\implies \{\mathtt{m}_3, \mathtt{m}_4\} \rightarrow \{\mathtt{m}_2\} \end{aligned}$$

An implication $\mathtt{m}_i \rightarrow \mathtt{m}_j$ can be established between two attributes $\mathtt{m}_i, \mathtt{m}_j \in \mathtt{M}$ iff $\mu(\mathtt{m}_i) \leq_{\mathcal{K}} \mu(\mathtt{m}_j)$, e.g. in Figure 1.1 we have the implication $\mathtt{m}_5 \rightarrow \mathtt{m}_4$ (the concept labelled with $\mathtt{m}_5$ is "below" the concept labelled with $\mathtt{m}_4$). We can further generalize this idea as follows. If $\mathtt{C}_1$ is the most general formal concept containing attributes in $\mathtt{B}_1$, $\mathtt{C}_2$ is the most general formal concept containing attributes in $\mathtt{B}_2$, and $\mathtt{C}_1 \leq_{\mathcal{K}} \mathtt{C}_2$, then $\mathtt{B}_1 \rightarrow \mathtt{B}_2$. In the Example 2, we can see that the most general concept containing $\mathtt{B}_1$ is the one marked with $\mathtt{C}$ in Figure 1.1 which is a sub-concept of the most general concept containing $\mathtt{B}_2$, which in this case is $\mu(\mathtt{m}_2)$ or the one labelled with $\mathtt{m}_2$.

If an implication is established between $\mathtt{B}_1$ and $\mathtt{B}_2$ in both directions (i.e. $\mathtt{B}_1 \rightarrow \mathtt{B}_2$ and $\mathtt{B}_2 \rightarrow \mathtt{B}_1$) then we have an "equivalence relation" between them or a "definition", denoted as $\mathtt{B}_1 \leftrightarrow \mathtt{B}_2$. An equivalence relation can be established between two attributes $\mathtt{m}_i \leftrightarrow \mathtt{m}_j$ iff $\mu(\mathtt{m}_i) \equiv \mu(\mathtt{m}_j)$, e.g. in Figure 1.1 we have a definition between the attributes $\mathtt{m}_2$ and $\mathtt{m}_3$.

Similar to formal concepts, the *support* of an implication is given by:

$$\sigma(\mathtt{B}_1 \rightarrow \mathtt{B}_2) = \frac{|(\mathtt{B}_1 \cup \mathtt{B}_2)'|}{|G'|} \tag{1.10}$$

An association rule can be established between any two attribute sets such as $\mathtt{B}_1 \cap \mathtt{B}_2 = \emptyset$ (which we will denote as $\mathtt{B}_1 \Rightarrow \mathtt{B}_2$) to which a confidence measure is defined in Equation 1.11 [1].

$$conf(\mathtt{B}_1 \Rightarrow \mathtt{B}_2) = \frac{|(\mathtt{B}_1 \cup \mathtt{B}_2)'|}{|\mathtt{B}_1'|} \tag{1.11}$$

In the case of association rules $\mathtt{B}_1$ is called the *antecedent* and $\mathtt{B}_2$ is called the *consequent* of the rule.

Example 3. Consider the association rule $\{\mathtt{m}_2, \mathtt{m}_3\} \Rightarrow \{\mathtt{m}_4\}$ in the formal context of Table 1.1. The confidence of this rule is given by:

$$\begin{aligned} conf(\{\mathtt{m}_2, \mathtt{m}_3\} \rightarrow \{\mathtt{m}_4\}) &= \frac{|\{\mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4\}'|}{|\{\mathtt{m}_2, \mathtt{m}_3\}'|} \\ &= \frac{|\{\mathtt{g}_2, \mathtt{g}_3\}|}{|\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}|} \\ &= \frac{2}{3} \end{aligned}$$

In this case we say "66% of the objects containing $\{\mathtt{m}_2, \mathtt{m}_3\}$ also contain $\mathtt{m}_4$".

Notice that an association rule is a generalization of the idea of implication. It is easy to show that an association rule with confidence equal to one is an implication. As an example, consider the association rule $\{\mathtt{m}_3, \mathtt{m}_4\} \Rightarrow \{\mathtt{m}_2\}$ with confidence:

$$\begin{aligned} conf(\{\mathtt{m}_3, \mathtt{m}_4\} \rightarrow \{\mathtt{m}_2\}) &= \frac{|\{\mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4\}'|}{|\{\mathtt{m}_3, \mathtt{m}_4\}'|} \\ &= \frac{|\{\mathtt{g}_2, \mathtt{g}_3\}|}{|\{\mathtt{g}_2, \mathtt{g}_3\}|} \\ &= \frac{2}{2} = 1 \end{aligned}$$

$$\sigma(\{\mathtt{m}_3, \mathtt{m}_4\} \rightarrow \{\mathtt{m}_2\}) = \frac{|\{\mathtt{g}_2, \mathtt{g}_3\}|}{|\{\mathtt{g}_2, \mathtt{g}_3\}|}$$

We can observe that this corresponds to the implication rule in the previous example. Association rules are of special interest in itemset mining where they can be translated to sentences like "90% of the customers buying items 1 and 2 also buy item 3". In the context of association rules, it can be said that *confidence* measures the rule's strength while the *support* provides a notion of statistical significance [1].

1.2.2 Conceptual Stability

Concept stability [84] was introduced as a measure for assessing the randomness associated to a data sample and the hypothesis that can be derived from that sample. In [125], the "intensional stability" of a concept is defined as "the probability of preserving its intent after leaving out an arbitrary number of objects". This notion makes stability a measure of "noise", this is if a formal concept actually represents reality or if it is built by chance as an artefact of randomness.

In order to provide a more "intuitive notion" of what stability is, let us consider the following scenario. You are in charge of deciding which interest groups have to be financially supported by the University. To do so, you decide that the interest groups to be supported should be those with more probabilities to remain even if individual members of the group left it, thus we will assume that a group will break if the common interest of all its members it is not the topic of the group. To simplify the problem, let us consider only two groups, the "Club of football" with three members and the "Club of arts", with four members. Then, you run a little pool about the interests of each individual member in both clubs, answers are given in Table 1.2.

With respect to the "Club of football", we can see that two members have interest in sports in general, while only one of them is interested in football. We can infer from this that F.2 and F.3 would not be only interested in football-related activities if F.1 was not in their group, and if the latter were to leave it, F.2 and F.3 would quickly change the group name to "Club of sports". This would not be the case if either F.2 or F.3 (or both) left the group, since F.1 is just interested in football.

For the case of the "Club of arts", the case is a little more complicated. All the members are interested in different arts which is the only thing they all have in common. If A.4 were to leave, the group would probably remain as a club of arts since there are still mixed interests related to arts within the group. However, if A.1 and A.4 were to leave, A.2 and A.3 would probably

Individual	Interest
F.1	Football
F.2	Sports
F.3	Sports
A.1	Painting
A.2	Painting & Music
A.3	Music & Sculpture
A.4	Sculpture & Painting

Table 1.2: Answers given by the members of the "Club of Football" (F.1,F.2,F.3) and members of the "Club of arts" (A.1,A.2,A.3,A.4)

Remaining members	Group remains
F.1,F.2,F.3	✓
F.1,F.2	✓
F.1,F.3	✓
F.2,F.3	✗
F.1	✓
F.2	✗
F.3	✗
$\emptyset$	✗

Table 1.3: Different possible scenarios for the "Club of football"

create a "Club of music", since they both agree in that particular interest. A similar thing would happen if A.3 were to leave, since A.1, A.2 and A.4 would probably create a "Club of painting".

Tables 1.3 and 1.4 shows all possible scenarios for both clubs, including two trivial ones, the case where nobody leaves (first row) and the case where everybody leaves (last row). We can see that in the case of the "Club of football", in half of the cases the group remains, while in the case of the "Club of arts" in less than one third of the cases the group remains. In this case we would say that the "Club of football" is *more stable* than the "Club of arts" (leading to the decision of financing it).

As we have seen in the example, stability has to do with the probability of a formal concept to maintain its intent (the topic of the club) if we remove some elements from its extent (some members leave). This is called "intensional stability" and, given a formal concept $(\mathtt{A}, \mathtt{B})$, it is defined as in Equation 1.12.

$$\sigma(\mathtt{A}, \mathtt{B}) = \frac{|\{\mathtt{C} \subseteq \mathtt{A} \mid \mathtt{C}' = \mathtt{B}\}|}{2^{|\mathtt{A}|}} \tag{1.12}$$

1.2.3 Many-valued Contexts

Certain attributes are not Boolean but take different values (consider the attribute "age" of a person which may take a number between 0 and 120 or "genre" of a film which may take values within a set of possibilities, e.g. "action", "comedy", etc.). This kind of attributes is called "many-valued attributes" and their relation with objects is represented in a many-valued context.

Consider a set of many-valued attributes $\mathtt{M}$. For a given attribute $\mathtt{m} \in \mathtt{M}$, the set of all values that an object may take for it, is denoted by $\mathtt{W}_\mathtt{m}$. Then, $\mathtt{W}$ is the set of values that objects in $\mathtt{G}$

Remaining members	Group remains	New Club
A.1,A.2,A.3,A.4	✓	-
A.1,A.2,A.3	✓	-
A.1,A.2,A.4	✗	Painting
A.1,A.3,A.4	✓	-
A.2,A.3,A.4	✓	-
A.1,A.2	✗	Painting
A.1,A.3	✓	-
A.1,A.4	✗	Painting
A.2,A.3	✗	Music
A.2,A.4	✗	Painting
A.3,A.4	✗	Sculpture
A.1	✗	Painting
A.2	✗	Painting & Music
A.3	✗	Music & Sculpture
A.4	✗	Sculpture & Painting
$\emptyset$	✗	

Table 1.4: Different possible scenarios for the "Club of arts"

	Language	Pub. Year	Genre	Author
Divine Comedy	Italian	1555	Epic	Allighieri
Les Misérables	French	1862	Epic	Victor Hugo
Don Quixote	Spanish	1605	Epic	Cervantes
Hamlet	English	1603	Tragedy	Shakespeare
La Araucana	Spanish	1569	Epic	Ercilla
Les Trois Mousquetaires	French	1844	Novel	Dumas
Decamerone	Italian	1886	Novel	Bocaccio
War and Peace	Russian	1869	Novel	Tolstoi

Table 1.5: Many-valued context example (information extracted from Amazon, Wikipedia and Google Knowledge Graph)

may take for attributes in $\mathtt{M}$. The ternary relation $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M} \times \mathtt{W}$ contains triples $(\mathtt{g}, \mathtt{m}, \mathtt{w}) \in \mathtt{I}$ indicating that object $\mathtt{g}$ takes the value $\mathtt{w}$ for attribute $\mathtt{m}$, or what is the same $\mathtt{m}(\mathtt{g}) = \mathtt{w}$. An object is restricted to take a single value per attribute, i.e. $(\mathtt{g}, \mathtt{m}, \mathtt{w}_1) \in I$ and $(\mathtt{g}, \mathtt{m}, \mathtt{w}_2) \in I$ iff $\mathtt{w}_1 = \mathtt{w}_2$. A many-valued formal context is defined by the tuple $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$. Table 1.5 shows a many-valued context built for eight different classic books where $genre(Quixote) = Epic$.

1.2.4 Conceptual Scaling

Given a many-valued context, it is possible to derive a formal context from it by the process of *conceptual scaling*. Formally, a scale $\mathcal{S}_m$ is defined for a many-valued attribute $\mathtt{m} \in \mathtt{M}$ as $\mathcal{S}_\mathtt{m} = (\mathtt{G}_\mathtt{m}, \mathtt{M}_\mathtt{m}, I_\mathtt{m})$, with $\mathtt{m}(\mathtt{G}) \subseteq \mathtt{G}_\mathtt{m}$. Different types of scaling are possible depending on the nature of the data (numerical, tree, sets) or the kind of application required. Often, we will define a scaled formal context derived from a many-valued formal context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$ as the triple $(\mathtt{G}, \mathtt{N}, \mathtt{J})$, where $\mathtt{N} \subseteq \mathtt{M} \times \mathtt{W}$. The description of J will define the type of scaling applied. For instance, when applying *plain scaling*, we will define $\mathtt{gJ}(\mathtt{m}, \mathtt{w}) \iff \mathtt{m}(\mathtt{g}) = \mathtt{w}$, e.g. plain scaling applied to

	Language					Genre			Pub.Year							
	Italian	Spanish	English	French	Russian	Epic	Tragedy	Novel	1555 $\vee\vert$	1569 $\vee\vert$	1603 $\vee\vert$	1605 $\vee\vert$	1844 $\vee\vert$	1862 $\vee\vert$	1869 $\vee\vert$	1886 $\vee\vert$
Divine Comedy	×					×			×							
La Araucana		×				×			×	×						
Hamlet			×				×		×	×	×					
Don Quixote		×				×			×	×	×	×				
Les Trois Mousquetaires				×				×	×	×	×	×	×			
Les Misérables				×		×			×	×	×	×	×	×		
Decamerone	×							×	×	×	×	×	×	×	×	
War and Peace					×			×	×	×	×	×	×	×	×	×

Table 1.6: Formal context after plain and nominal scaling from many-value context in Table 1.5

attributes *Language* and *Genre* in Table 1.6. For a numerical attribute (or an attribute with a given order $\leq$), its *ordinal scaling* will be given by $\mathtt{gJ}(\mathtt{m},\mathtt{w}) \iff \mathtt{m}(\mathtt{g}) \leq \mathtt{w}$, e.g. ordinal scaling applied to attribute *Pub. Year* in Table 1.6. Notice that in the case of attribute *Pub. Year* we could have defined arbitrary values for the scales (e.g. ≤ 1700 and > 1700). We provide the full ordinal scaling for the sake of generality.

1.2.5 Calculating a concept lattice

Several algorithms have been proposed to calculate, from a given formal context, the set of all formal concepts with its associated concept lattice structure [87]. Furthermore, different implementations are freely available online for most of these algorithms (e.g. Conexp-ng [9], ToscanaJ[10], Coron[11], Galicia[12], etc.). For all the experiments in this thesis we have developed our own system called Sephirot[13] which implements the AddIntent algorithm for concept lattice calculation [139].

1.2.6 Pattern structures

Pattern structures are a generalization of formal concept analysis designed to deal with complex object descriptions, i.e. when an object is not associated to a set of attributes [62]. Several types of pattern structures have been proposed in the literature depending on the nature of the object description, including:

- Graph pattern structures (original application) [62]
- Interval pattern structures [79]
- Partition and tolerance blocks pattern structures [9]

[9] https://github.com/fcatools/conexp-ng
[10] http://toscanaj.sourceforge.net
[11] http://coron.loria.fr
[12] http://www.iro.umontreal.ca/~galicia/
[13] https://code.google.com/p/sephirot/

In the pattern structure framework, a Galois connection is defined between the set $\wp(\mathtt{G})$ (i.e. the powerset of $\mathtt{G}$) and a semi-lattice of object descriptions denoted as $\underline{\mathtt{D}}$ (instead of between $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ such as in standard FCA). Thus, it is necessary to define the space of all object descriptions $\mathtt{D}$ to which an object is mapped through a function $\delta : \mathtt{G} \to \mathtt{D}$. Thus, for a given object $\mathtt{g}$, we denote its correspondent description as $\delta(\mathtt{g})$. For any two objects $\mathtt{g}_1, \mathtt{g}_2$, we define a similarity operator $\sqcap$ between their descriptions as $\mathtt{d} = \delta(\mathtt{g}_1) \sqcap \delta(\mathtt{g}_2)$, where $\mathtt{d} \in \mathtt{D}$ is called a "pattern". Patterns are built by the combination of object descriptions through the similarity operator and may (or may not) correspond to an actual object description.

Given any two patterns $\mathtt{d}_1, \mathtt{d}_2 \in \mathtt{D}$, we define the order between them ($\sqsubseteq$) w.r.t. the similarity operator as shown in Equation 1.13. The space of object descriptions (or indistinctly, the set of patterns) with the order defines a semi-lattice of object descriptions $\underline{\mathtt{D}} = (\mathtt{D}, \sqsubseteq)$.

$$\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1 \tag{1.13}$$

Analogously to FCA, we will define the derivation operators $(\cdot)^{\square}$ for a set of objects $\mathtt{A} \subseteq \mathtt{G}$ and a pattern $\mathtt{d} \in \mathtt{D}$ as described in Equations 1.14 and 1.15.

$$\mathtt{A}^{\square} = \prod_{\mathtt{g} \in \mathtt{A}} \delta(\mathtt{g}) \tag{1.14}$$

$$\mathtt{d}^{\square} = \{\mathtt{g} \in \mathtt{G} \mid \mathtt{d} \sqsubseteq \delta(\mathtt{g})\} \tag{1.15}$$

Finally, the pair $(\mathtt{A}, \mathtt{d})$ satisfying $\mathtt{A}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{A}$, is called a pattern concept. Pattern concepts can be ordered by extent inclusion yielding a pattern concept lattice in the same way that formal concepts do.

Pattern Structures for the rest of us

To illustrate how pattern structures are a generalization of FCA, in the following we will explain the "Set pattern structure" which frames FCA under the definitions given above. Incidentally, the conception of FCA in this manner corresponds to the original model design for a standard information retrieval system [105].

Consider the formal context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ in Table 1.7. We will define a "set pattern structure" as a subset of $\mathtt{M}$ and thus, the search space is given by the powerset of $\mathtt{M}$ (denoted as $\wp(\mathtt{M})$) represented in a lattice structure in Figure 1.2.

Let us define a mapping $\delta : \mathtt{G} \to \mathtt{D}$ assigning for each object $\mathtt{g} \in \mathtt{G}$, its correspondent description in $\mathtt{D}$. In this case, we will use the same definition for the object intent as described in Equation 1.4.

$$\delta(\mathtt{g}) = \{\mathtt{m} \in \mathtt{M} \mid \mathtt{gIm}\}$$

Thus, the assignation function δ can be derived directly from Table 1.7 (e.g. $\delta(\mathtt{g}_1) = \{\mathtt{m}_1, \mathtt{m}_2\}$). Figure 1.3 represents each single assignation. Let us define now a similarity operation for two given object representations in $\mathtt{D}$. Let $\mathtt{g}_i, \mathtt{g}_j$ be two objects in $\mathtt{G}$, we have:

$$\delta(\mathtt{g}_i) \sqcap \delta(\mathtt{g}_j) = \{\mathtt{m} \in \mathtt{M} \mid \mathtt{g}_i\mathtt{Im} \text{ and } \mathtt{g}_j\mathtt{Im}\} = \delta(\mathtt{g}_i) \cap \delta(\mathtt{g}_j)$$

It is easy to observe that Equation 1.13 holds for $\sqcap$ as above defined when $\sqsubseteq \equiv \subseteq$, for example:

$$\delta(\mathtt{g}_1) \sqcap \delta(g_2) = \{\mathtt{m}_1, \mathtt{m}_2\} = \delta(\mathtt{g}_1) \implies \delta(\mathtt{g}_1) \sqsubseteq \delta(\mathtt{g}_2)$$

Where $\delta(\mathtt{g}_1) \subseteq \delta(\mathtt{g}_2)$. Finally, for the derivation operators defined in Equations 1.14 and 1.15, it is clear that the former coincides with the definition in Equation 1.1 given that $\sqcap \equiv \cap$. As for the latter, the fact that definitions in Equation 1.1 and 1.15 are equivalent can be derived as follows:

$$\begin{aligned}
\mathtt{d}^{\square} &= \{\mathtt{g} \in \mathtt{G} \mid \mathtt{d} \sqsubseteq \delta(\mathtt{g})\} \\
&= \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{d}, \mathtt{m} \in \delta(\mathtt{g})\} \\
&= \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{d}, \mathtt{m} \in \{\mathtt{m} \in \mathtt{M} \mid \mathtt{gIm}\}\} \\
&= \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{d},\ \mathtt{gIm}\} \\
&= \mathtt{d}'
\end{aligned}$$

Thus, a set pattern concept $(\mathtt{A}, \mathtt{d})$ derived from the set pattern structure $(\mathtt{G}, \mathtt{D}, \delta)$ (or $(\mathtt{G}, (\wp(\mathtt{M}), \sqsubseteq), \delta)$) is such that, given a formal concept $(\mathtt{A}, \mathtt{B})$ of formal context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$, then $\mathtt{B} = \mathtt{d}$. Furthermore, there are exactly as many intents in $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ as there are set pattern structures in $(\mathtt{G}, \mathtt{D}, \delta)$ and their concept lattice representations are isomorphic. Figure 1.4 shows the set pattern structure lattice calculated for the running example. Each dashed line takes a set of objects to their set pattern representation. Consider that the right lattice is Boolean, thus not every subset of $\mathtt{M}$ has a pre-image in the left lattice of extents.

1.3 Information Retrieval

Information Retrieval (IR) is a research domain that deals with the indexing and retrieval of information given a user request translated from his information needs [93, 6]. We identify IR with a research domain for the sake of generality, even though it can refer to a paradigm of thought, to actual systems (computational or not) or to a set of techniques.

However, in order to land it into a set of concepts we can manage, let us define an IR system as a process through which information in the form of documents can be organized so users can access it. Thus, we can split an IR system in two main processes, namely Indexing and Retrieval as depicted in Figure 1. Indexing refers to the process through which a set of documents is taken to a representational space (which we will call *index*) by the use of a *representation function*. Examples for indices may be a lattice or a vector space. Retrieval refers to the process through

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$
$\mathtt{g}_1$	×	×		
$\mathtt{g}_2$	×	×	×	
$\mathtt{g}_3$		×	×	×
$\mathtt{g}_4$	×		×	×

Table 1.7: Formal Context Example 2

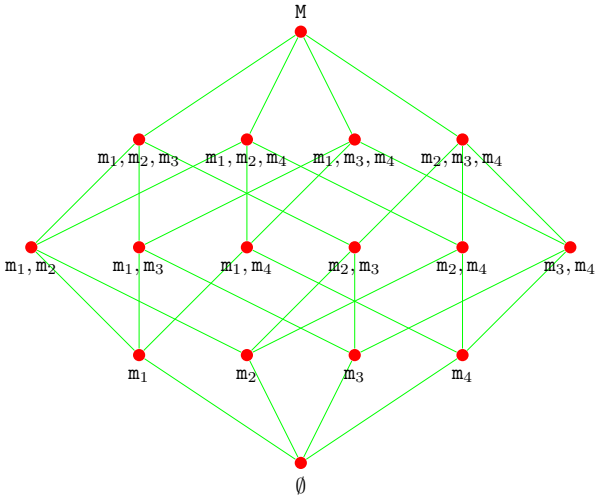


Figure 1.2: The powerset $\wp(\mathtt{M})$ represented as a lattice

which a user request (which we call *query*) is taken to the index by a *representation function* and matched with document representations through a *retrieval function* used to obtain documents.

Depending on the retrieval function and the kind of representational space used, different IR systems can be defined, for example the Boolean Retrieval model which uses dictionaries and inverted indices, or the vector space model which represents documents as coordinates in an arbitrary set of dimensions. In the following, we define the Boolean Retrieval model which is used through this and next chapters. The vector space model is defined as part of the theoretical background of Chapter 3.

The Boolean Retrieval Model

The Boolean retrieval model is considered as the first and one of the simplest techniques to index and retrieve documents [6, 93]. Given a collection of documents $\mathtt{G}$, we consider each document $\mathtt{g}$ as represented by a *conjunction* of Boolean descriptors $\mathtt{g}' \subseteq \mathtt{M}$, where $\mathtt{M}$ is the set of all descriptors (sometimes called "repertory" or "dictionary"). A query ("request", or "prescription") is defined as a set of descriptors connected by a logical operator AND, OR, NOT. Traditionally, descriptors in the Boolean IR model are terms within documents.

It is worth noticing that in this description we have jumped through many well-defined procedures in IR. For example, the operation to obtain the descriptors of a given document that we naturally associate with the derivation operator defined in FCA ($\mathtt{g}'$), in IR is modelled through an *inverted index* which associates dictionary entries (descriptors) with postings (documents). Furthermore, in our setting we consider documents as already described by a set of conjunctive descriptors. In reality, documents are built from unstructured text and several techniques are available to obtain a reduced set of descriptors from it (e.g. parsing, tokenizing, lemmatization, disambiguation, etc. [93]). All these procedures are by no means irrelevant. In fact, they can be considered critical to the effectiveness of the IR tool they support. Nevertheless, they are not a part of our model and we consider them as given, i.e. we consider that document corpora are already in a document-descriptor format. This is actually true for many of the test datasets available and for all we use through this document. We will provide a further description for some of them in Section 2.4, Chapter 2.

The simplest query is given by a set of descriptors connected by AND and is called a "con-

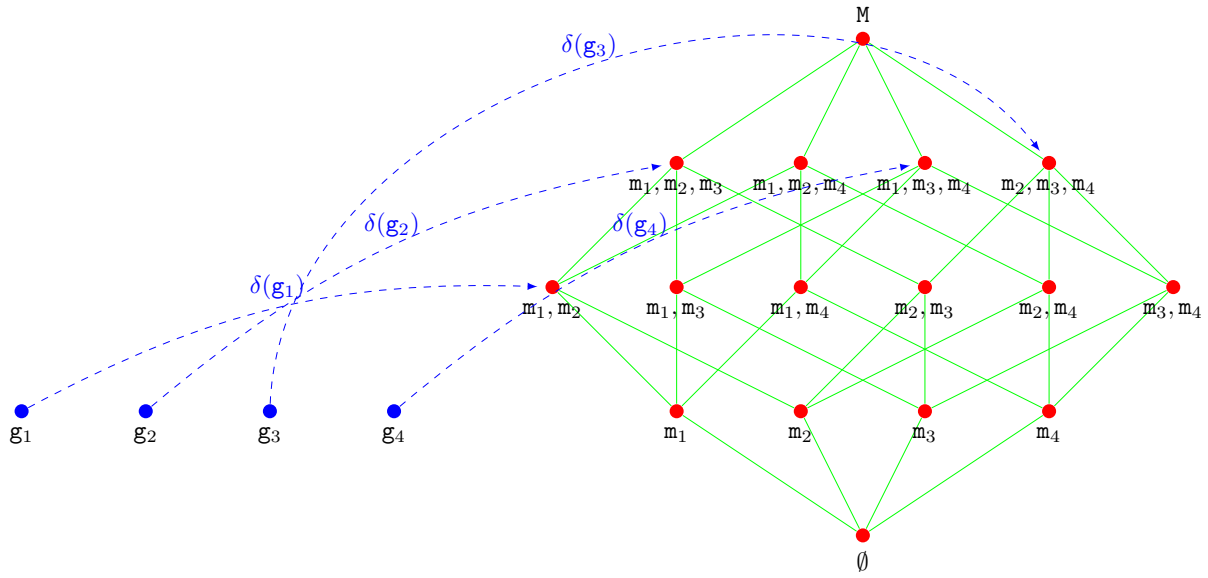


Figure 1.3: Mapping $\delta : \mathtt{G} \rightarrow \mathtt{D}$

junctive query". Given a conjunctive query $\mathtt{q}_{and}$, the set of relevant documents to be retrieved ($\mathtt{q}'_{and}$) are those that contain *at least* all the descriptors in the query. A disjunctive query (using *OR*) can always be split into its conjunctive parts and the set of relevant documents can be computed by the union of each separate set of relevant documents. A similar approach can be applied for *NOT*. In this work we will consider every query $\mathtt{q}$ as being conjunctive, unless indicated otherwise.

1.4 A survey on Formal Concept Analysis based Information Retrieval approaches

Surveying the intersection of Formal Concept Analysis (FCA) [63] and Information Retrieval [6] is not an easy task. The main complexity is that both domains have an application range so wide that just getting a relevant set of articles to report about is a knowledge discovery process in itself. This is clearly exemplified by the survey presented by Poelmans et al. in 2012 [118] where FCA is used to report about 103 articles related to topics of FCA and IR in a period of only six years (2003-2009) crawled from the Web. In this chapter we intend to approach the surveying in a more general and integral manner. We try to answer a very simple question. How FCA and concept lattices have been used in the context of IR applications? We answer this in a chronological narration, trying to cover the last 25 years of research since the first inception of the use of lattice structures to model the space of possible queries (or prescriptions, as they were called) to the last approaches, supporting file systems and semantic technologies.

As we can observe, most of the approaches presented here rest over a limited pool of *ideas and techniques* associated with FCA/IR but applied to a myriad of domains and applications. These ideas are:

1. Using a concept lattice as a model of the description and document spaces

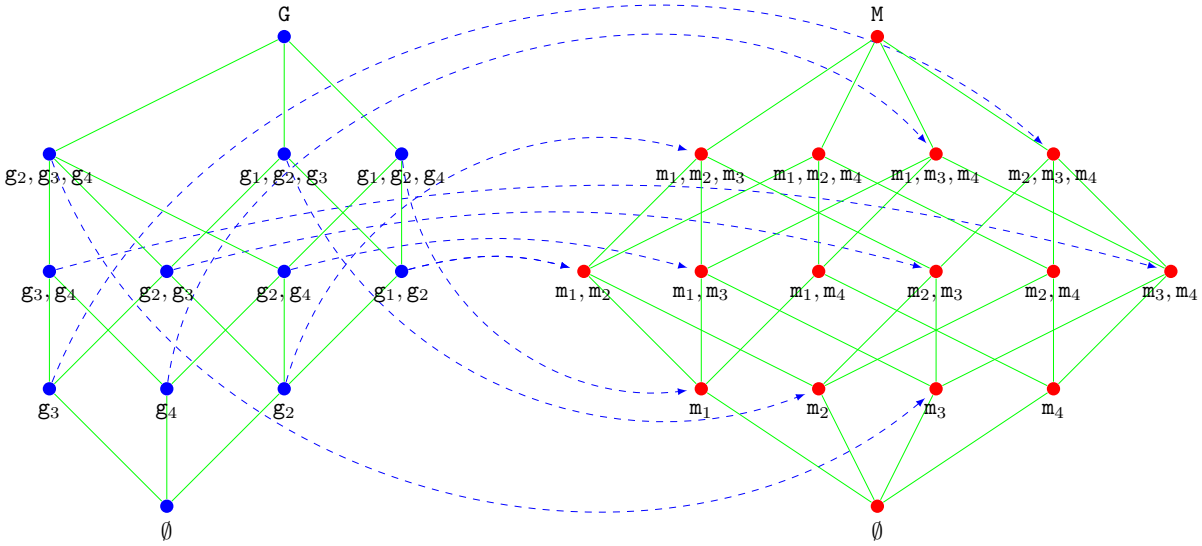


Figure 1.4: Set pattern structure derived from Table 1.7

2. Enriching the description space through external knowledge sources
3. Enabling Relevance Feedback
   - Mixing querying and browsing
   - Query-by-navigation
   - Query-by-example
4. Using a concept lattice as a support for automatic retrieval

Our goal in this survey is two-fold. Firstly, we want to catalogue these ideas so future endeavours may have an easier way reaching further domains while developing new different and more interesting techniques. Secondly –and perhaps more important– we set the theoretical background in which the rest of this thesis is rooted.

Other surveys

Along with the work of Poelmans [118], there have been other important reviews of the literature regarding FCA and IR [26, 122, 138]. In 2005, Carpineto and Romano [26] described the main possible tasks that FCA could perform regarding querying and indexing by summarizing some of their work in the field. In 2007, Uta Priss [122] dedicated a full chapter to describe the state-of-the-art up to 2004 on FCA-based IR in her paper on *FCA and Information Sciences*. The last of these reviews was presented by Valverde and Peláez-Moreno in 2013 in the first (and sadly, the last) workshop on *Formal Concept Analysis meets Information Retrieval* in the context of the European Conference on Information Retrieval (ECIR 2013)[14]. This work differentiates

[14] `http://fcair.hse.ru`

between what is FCA *in* IR and what is FCA *for* IR, the latter of which refers to the possibility of *"augmenting IR with the methods and ideas of FCA"*. The authors describe these ideas in seven "affordances" of FCA for IR, classifying with them the body-of-work of FCA-based IR approaches.

All of these works are of maximum relevance to any researcher interested in retrieval using FCA. Our work differentiates in two aspects. Firstly, the benefit of time enables us to update the list of reported works. Secondly, we approach the surveying task in a more didactic manner. Instead of classifying works by an arbitrary criterion, we tell the story of how these works emerged in the IR arena and how they diverged given the natural characteristics of FCA.

1.4.1 Pre-FCA history - A lattice to model the description and document spaces

Lattice structures were early adopted by information scientists as a model of document indexing [54, 105]. As early as 1956, Robert A. Fairthorne [54] discussed how to model a library classification system by producing all possible requests (queries) as combinations of categories (descriptors) and logical connectors (AND, OR, NOT) and how this model could be compared to a "free distributive lattice". Some years later, Calvin Mooers [105] would consider two spaces for this model, namely the space of prescriptions P (descriptors) and the space of all possible documents subsets as $L = \wp(G)$.

He realised that L with the set inclusion operator $\subseteq$ was naturally a partially-ordered set (or poset) and that, under certain circumstances (actually when $P = \wp(M)$), P could also be modelled as such. With this, a retrieval system consists in a transformation $T : P \to L$ that is able to take a prescription (query) into the largest subset of documents that satisfies it (see Figure 1.5).

It is important to note that Mooers did not describe an actual IR system, but a "model" for retrieval systems that would enable the comparison of different approaches. Then, we can observe that FCA is an instance of this model, where the transformation T is naturally represented by a Galois connection defined between $\wp(G)$ and $\wp(M)$ and where the concept lattice is an elegant solution for the spaces P and L as it represents them in an integrated manner. Particularly, when this Galois connection is defined in terms of the derivation operator ($(\cdot)'$), FCA becomes an implementation of the Boolean IR model.

The underlying model of FCA-based IR approaches

Let us introduce a general model of Boolean retrieval using the FCA framework with an example. In the following sections, we will re-use this model to explain how the tasks of browsing and querying can be performed using a concept lattice. Consider a formal context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ of documents and descriptors as the one shown in Table 1.8. Documents for a query $Q \subseteq M$ are retrieved through the derivation $Q' \subseteq G$ which works as the "transformation" T shown in Figure 1.5. For example, the conjunctive query $Q = \{arthroscopy, complication\}$ has as answer documents in $Q' = \{\mathtt{g}_7, \mathtt{g}_8\}$.

Key aspects: The query Q can be naturally *extended* to Q'', which of course, contains the same set of answers Q'. In the example, the query $Q = \{MRI\}$ extends to $Q'' = \{MRI, medicine\}$ and they both have the same answer $Q' = \{\mathtt{g}_3, \mathtt{g}_4\}$. This fact was already discussed by Mooers [105] and has been exhaustively exploited by FCA-based IR approaches to provide context to user queries (in the example, showing the user that his answer for MRI is within a *medical* context

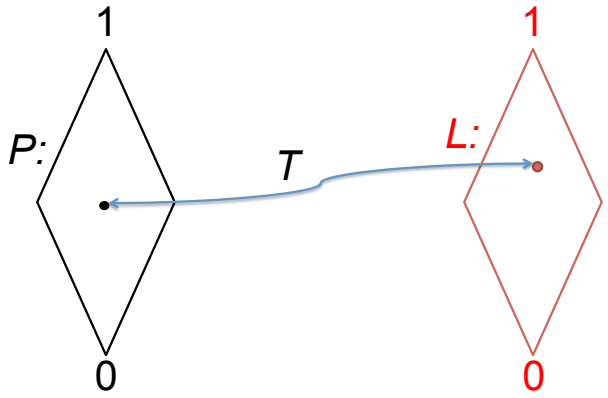


Figure 1.5: Mooers' model: *"The space P of all possible retrieval prescriptions (queries), the space L of all possible document subsets, and the retrieval transformation T associating points in P with points in L."*

instead of several other possible interpretations[15]). The formal concept formed by (Q', Q'') has been called virtual node, virtual concept or *query concept*, and represents both, the extended query (intent), and the set of retrieved documents (extent). Notice that the latter can be an empty set if there are no documents satisfying the query (hence the name *virtual*). Finally, in this chapter we will make the distinction between "query extension" and "query expansion". The first of which refers to the *closure* of the query w.r.t. $(\cdot)'$. The second refers to an actual *modification* of the query by taking a set Q_1 where $Q_1'' \neq Q''$ and in general $Q_1 \cap Q \neq \emptyset$ (i.e. finding a query Q_1 related to Q which yields different results).

Throughout all the approaches discussed in this chapter, the underlying model described above has not varied much (notice that the book of Barbut and Monjardet which included what will be FCA later was published in 1970! [10, 140]). This fact is in no way a negative point for FCA-based IR approaches, but actually a statement about the adequacy of the model to fit in different tasks and domains. On the other hand, this advantage of FCA is also one of its main drawbacks when dealing with modern IR systems.

The Boolean IR model was quickly considered too limited for the complex tasks involved in the retrieval of documents considering the size of modern document collections or the nature of their descriptions (e.g. numeric instead of Boolean). The IR community would shift to more complex models such as the vector space model (for ranking documents by "relevance" w.r.t. a query) or the probabilistic model (for predicting which are those more "relevant" for a user). Current introductory books on IR [93, 6] do not mention lattice structures (not to say concept lattices) as valid IR models[16]. In [93], the chapter on the Boolean IR model finishes with the following quote attributed to Calvin Mooers in a book of Robert A. Fairthorne (1961):

> *"It is a common fallacy (...) that the algebra of George Boole (1847) is the appropriate formalism for retrieval system design. This view is widely and uncritically accepted as it is wrong."*

[15] `http://en.wikipedia.org/wiki/MRI_(disambiguation)`

[16] Actually, in [6] there is an entry of two paragraphs - in a 500 pages book - about lattices in chapter 10 about user interfaces and visualization, referencing [22, 113] as systems for query reformulation (expansion).

	patient	laparoscopy	scan	user	medicine	response	time	MRI	practice	complication	arthroscopy	infection
g_1	×	×	×							×		
g_2			×	×	×	×	×		×			
g_3		×		×	×			×				
g_4	×				×			×				
g_5				×		×	×					
g_6									×		×	
g_7										×	×	
g_8										×	×	×
g_9											×	×

Table 1.8: A document-term formal context

A commentary on Mooers' quote

At this point in history, it is hard to argue with the last quote on the effectiveness of the Boolean retrieval model. Modern retrieval systems cannot rely on this paradigm for several reasons including system scalability, user interaction and document representation (consider non-textual documents such as images). Nevertheless, Boolean retrieval became the "de facto" standard of the industry for several years following the date of Mooers' statement, leading to interesting developments such as the Extended Boolean Retrieval model [128] and most of those described in the next sections. This was despite the fact that the vector space model had been available [129] since 1975.

Indeed, the Boolean retrieval is not *the* appropriate formalism for retrieval system design as the hammer and nails are not *the* appropriate tools for sticking two pieces of wood together. As in any toolbox, the Boolean retrieval model takes part in a set of different tools which vary in capabilities and applications. It is up to us, the builders, to know when our tools work the best.

1.4.2 FCA meets IR

The bad scenario for the Boolean retrieval model and its drawbacks did not stop many researchers from developing different applications using this paradigm. In the '90s, the first FCA-based IR systems were developed, while several other systems based on the use of lattice structures became popular.

Non-FCA lattice-based IR systems

Pedersen in [113] introduced BRAQUE (BRowse And QUery Environment) as a system that allowed the navigation of a document collection modelled as a *relationship lattice* [114], strongly resembling the features of a concept lattice. At the AT&T Bell labs, Ginsberg [64] introduced WorldViews, consisting *"of a system for automatic document indexing, an information retrieval system and a user interface"* using a taxonomy modelled as a lattice structure. In the work of Bosman et al. [15], a similar approach to Ginsberg's WorldLattice was presented for creating a "Hyperindex" of a *faceted hierarchical thesaurus* using a lattice structure. The lattice supported

a "query-by-navigation" approach where the user could "refine" or "enlarge" a query. In the domain of software engineering, Mili et al. [102] proposed a lattice-based index of software descriptions for retrieval purposes based on software reuse needs. The authors describe two types of retrieval namely, *"exact"* which resembles the Boolean retrieval model and "approximate" measuring "proximity" w.r.t a given query.

FCA-based IR systems

It was the proposition of Godin et al. [66] that revealed the main capabilities of concept lattices for indexing and retrieval as an alternative to Boolean querying and hierarchical classifications. This work was built over the initial user interaction design proposed by the same authors years before [67, 65] . A major highlight in this work is the efficient browsing capabilities generated from a document collection by the construction of a concept lattice which actually represents a query space. In this manner, the user can pose different queries without explicitly indicating a set of terms to be sought within documents. An important advantage of this model is that users do not have to be completely familiarised with the lexicon used for indexing.

In the same year, Carpineto and Romano presented their system GALOIS [19] for conceptual clustering[17] which would be later implemented for information retrieval purposes through a query browsing interface called ULYSSES [20, 22]. ULYSSES develops further in the model for the unification of querying and browsing plus a third procedure called "bounding". The latter allows the user to restrict the search space within the concept lattice (deriving a sub-lattice) by including into the query sentences such as "all documents indexed by a given term m" (i.e. contained in formal concepts $(A, B) s.t. (A, B) \leq (m', m'')$) and "all documents not indexed by a given term m" (i.e. contained in formal concepts (A, B) such that $B \cap m'' = \emptyset$). Experimentation showed similar results to a plain Boolean retrieval system.

As Fairthorne proposed [54], in an ideal world we could take the descriptions of all the documents in a library and create a map of all the possible requests that could be made (this map would be the P space in Figure 1.5). However, this is a rather an unlikely scenario as the size of such map grows *"faster than exponentially"* w.r.t. the number of categories [121]. Instead, we would prefer to generate a smaller P space that represents the "most meaningful" queries[18]. For this reason, two main strategies were embraced. Firstly, the use of an authoritative source such as the thesaurus-based WordLattice in [64] which would model in a more concise manner the space P. Secondly, the elicitation of this space from document features (lexical properties [15], metadata [113] or terms [66, 20]).

An anecdote. Mooers described the size of the search space of a document collection (L in Figure 1.5) of one million documents as the number of subsets we can construct from it, being the staggering figure of $10^{310,000}$ [105]. This remembered one of the authors the description of a googol (10^{100}), a number proposed in 1938 by mathematician Edward Kasner to exemplify the difference between "an unimaginably large number and infinity" [19]. While a googol is much larger than the number of particles in the observable universe[20], we can see that the L space is much larger than a googol. Apparently, we were not the first ones to step on this interesting fact. In 1997, a couple of entrepreneurs looking for a name for their search engine, in an attempt

[17] Actually, GALOIS is an incremental algorithm for building a concept lattice.

[18] It is worth mentioning that the "meaningfulness" of a request is a matter of perspective. What is meaningful in a domain may not be in another. Meaning also changes with time.

[19] Wikipedia article - `http://en.wikipedia.org/wiki/Googol`

[20] Video about googol from the University of Nottingham - `https://www.youtube.com/watch?v=8GEebx72-qs`

to represent the "indexing of an immense amount of data" [21], registered the misspelled version "Google".

1.4.3 Enriching the description space through external knowledge sources

In the FCA-IR model explained in Section 1.4.1, attributes are descriptors obtained from the set of documents. As previously explained, this space (P) can be very large but other than that it can suffer from other problems. For example, it can be non-representative of the document set by different reasons (poor document description, poor vocabulary, incompleteness, etc.). Regarding these issues, it may be useful to use an external knowledge source to complement document descriptions. For example, if we are interested in considering synonymia for indexing (e.g. relating documents referring to *"concept lattices"* and *"Galois lattices"*) we may use a thesaurus. In case we are interested in considering hierarchical relations (e.g. relating documents referring to *"monkeys"* with those referring to *"primates"*) we may use a taxonomy. On the other hand, if we are interested in considering logical implications (e.g. relating documents written by a French author to those written by a German author using the label *"European literature"*) we may use an ontology.

With these concerns, in 1996 Carpineto and Romano proposed a modified version of the GALOIS system to include "background information" in the form of a thesaurus for document indexing using FCA [21]. The modification was made in the order relation between formal concepts ($\leq_{\mathcal{K}}$) using the order between document descriptors ($\leq_{\mathcal{T}}$) induced by a thesaurus as follows:

$$(A_1, B_1) \leq_{\mathcal{K}} (A_2, B_2) \iff \forall m_2 \in B_2, \exists m_1 \in B_1 \; s.t. \; m_1 \leq_{\mathcal{T}} m_2$$

Furthermore, they redefined the intersection between two descriptor sets as:

$$B_1 \cap^* B_2 = \{m_i \mid m_i \geq_{\mathcal{T}} m_1, m_2 \quad \text{with } m_1 \in B_1, m_2 \in B_2, m_i \in \mathcal{T}, \nexists m_j \in \mathcal{T}, s.t. \; m_i \geq_{\mathcal{T}} m_j \geq_{\mathcal{T}} m_1, m_2\}$$

From the example in Table 1.8, consider a thesaurus $\mathcal{T}$ with the relations *arthroscopy, laparoscopy* $\leq_{\mathcal{T}}$ *endoscopy*[22]. Then, $\{laparoscopy\} \cap^* \{arthroscopy\} = \{endoscopy\}$ and we can build the formal concept $(\{g_1, g_2, g_3, g_6, g_7, g_8, g_9\}, \{endoscopy\})$. Consider this analogous to including in the formal context the attribute *endoscopy* and the relation where each document related either to *laparoscopy* or *arthroscopy* is also related to *endoscopy*.

The authors argue that this approach would lower the complexity associated to computing the concept lattice compared to the more simple approach of adding the thesaurus terms to the initial formal context. In 1997, Uta Priss presented several propositions for a FCA-based IR system in which three main components were discussed [120]. Firstly, a combined formal context comprising document descriptors and other metadata components (e.g. publisher, author, etc.). Fields of this kind were coded by many-valued formal contexts which were later scaled (see attribute scaling in [63]). The second component described the inclusion of a thesaurus within the formal context by two approaches, namely by mapping document-descriptor pairs to thesaurus elements, and by constructing a combined formal context considering documents, descriptors

[21] David Koller on the origin of the name "Google" `http://graphics.stanford.edu/~dk/google_name_origin.html`

[22] Wikipedia categories `http://en.wikipedia.org/wiki/Category:Endoscopy`

and thesaurus elements in a relational concept analysis (RCA) manner (this RCA proposition is formally different from the one presented by Huchard et al. [71]). The third component referred to the use of "nested line diagrams" to represent in a better manner the combination of different concept lattices in an integrated view offering different description levels within a document collection.

Some of these ideas were later revisited by Cole and Eklund in 1999 [41] where the authors proposed an interactive e-mail retrieval system based on FCA. The formal context was built using "classifier outputs" as attributes which the user was asked to order in a hierarchy (`G` is a set of emails). *Conceptual scaling* was applied to many-valued attributes deriving views (sub-lattices) that were more manageable for the user to browse than the concept lattice of the entire email collection. In 2003, the authors (plus Gerd Stumme) would propose an extension of their work into a fully integrated system called "HIERMAIL" [42] in which nested-line-diagrams were used to represent conceptual scales (instead of sub-lattices) for knowledge discovery over an e-mail collection. Incidentally, Cole and Eklund had proposed a "folding" and "unfolding" mechanism (using the same notion of conceptual scales) for the concept lattice in a previous work oriented to model a document retrieval system in which documents were indexed by a medical thesaurus called SNOMED [40], although these procedures were not clearly defined.

A similar approach for domain-specific interactive FCA-based IR systems was presented by Mihye and Compton in 2001 [80] and later extended in [81]. An interesting point of this work is that it addresses the fact that taxonomies used to index documents are not static and should evolve through time. For this reason, the concept lattice is used not only to retrieve documents but also to aid users in the annotation of documents and in the evolution of the taxonomy.

1.4.4 Relevance Feedback and Automatic Retrieval

Relevance Feedback

Other than choosing and modelling the kind of data to be used as attributes in a formal context, an important factor in the efficiency of a retrieval system is to help the user closing what is usually called the knowledge or "the cognitive gap" [74]. The cognitive gap describes the distance between the space occupied by the actual information needs of a user and the space occupied by its ability to describe its information needs. For example, consider a user searching for "the book which they made a film about and a wizard appears on it". Somebody could answer "Is it about a girl, a lion, a tin man and a scarecrow?" to which the user may answer "No, there are some kids in a school". Then, the answer could be narrowed down to the 7 books of the "Harry Potter" saga. Here we can see that the cognitive gap can be represented as the distance between the initial query, possibly with the keywords `"book film wizard"`, which the user is able to provide, and the query that he actually needs to provide which is `"Harry Potter book"`.

In 1971, Rocchio proposed his famous *relevance feedback* model to overcome this issue [124]. We can understand relevance feedback as a "query calibration" system using extra user inputs. In the previous example, the initial user query was very abstract. Somebody (possibly the librarian), with knowledge about fantasy books asked the user a question based on the assumption that the answer may be "The wizard of Oz". The negative answer provides a feedback of relevance (i.e. "The wizard of oz" is not relevant) which is used to generate the query: `book film wizard school -"the wizard of oz"`[23].

In FCA terms, we can represent this scenario as the join of two object concepts as depicted in Figure 1.6b. The initial query yields concept 0 for which the system may propose concept 1

[23] In Google query syntax, '-' is used for excluding terms - `http://goo.gl/7RZrQl`

or concept 2.

This approach was proposed by Carpineto and Romano in 1998 through their system REFINER [23]. The user poses a query to which the system generates a "virtual concept in the lattice". By the use of the upper and the lower cover of the virtual concept, REFINER is able to propose minimal query refinements/enlargements (resp.) to the user. Experimental results showed significant better results in the search time employed by a user w.r.t. the Boolean IR model. In 2002, Grootjen et al. [68] proposed a similar rougher approach called "conceptual relevance feedback" further developed as a query expansion method [69] in the lines of pseudo-relevance feedback [93].

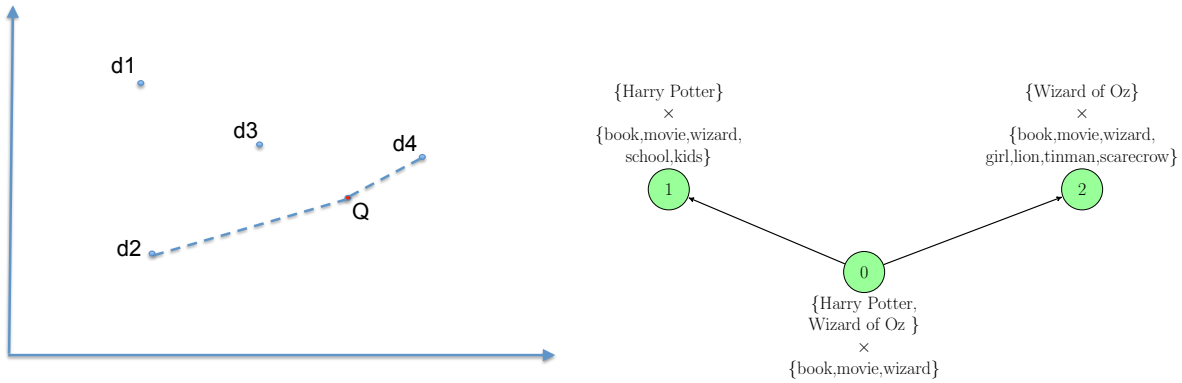
In 2007, Nauer and Toussaint [107] presented a model for "explicit relevance feedback"[24] over a standard Web search engine (such as Google) supported over a concept lattice. This model consisted of a constant iteration of the formal context by "extension" and "reduction" procedures. Extensions were made whenever the user submitted a new query or gave a positive assessment. Reductions were performed whenever the user gave a negative assessment. Similarly, explicit relevance feedback was also supported in a previous work by Martines and Loisant [96] for concept lattice-based image retrieval. Users were asked to evaluate images as "good" or "bad". An example of "implicit relevance feedback" can be found in the work of Ducrou et al. [50], supported by a procedure called "query-by-example". Instead of asking the user to give explicit relevance assessments, the query is modified by a sample set manually created by the user.

Ranking documents

So far we have reviewed approaches that assist the user in navigating the query space and deciding what *is* or *is not* relevant. This is usually achieved by providing an interface that helps them retrieve parts of the concept lattice by the use of "query-by-navigation", "query-by-browsing", "relevance feedback" or "query-by-example". This however is not what we are used to when dealing with search engines. The "file search program" of any operating system, or the mechanics of traditional Web search engines follows a very simple scenario. The user inputs a query and the system outputs a list of documents already ordered by the "relevance" w.r.t. that query. Thus, the system is provided with the notion of *what is relevant* and *what is not*. For instance, in the vector-space model, documents and queries are represented by points in an arbitrary Euclidean-space. "Relevance" in this case may be represented by the distance between a query and a document (the closer the document, the more relevant it is w.r.t. the query) as shown in the example of Figure 1.6a (an explanation on the meaning of the axis or why the documents and the query are located in the space as they are, is given later in this document, in Section 3.2.1. For more information see [93]).

A similar notion was adopted by Carpineto and Romano in 2000 in what they called concept lattice-based ranking (CLR) [24] for a fully automated retrieval system. Using the REFINER model, the virtual concept representing the query is placed in the concept lattice and a series of "concentric rings" around the virtual concept yields a distance that allows ranking documents (e.g. in Figure 1.6b, documents in concepts 1 and 2 - and not in concept 0 - are at distance 1, while documents in their super-concepts would be at distance 2). Different measures are also introduced in the work of Ducrou et al. [50] where instead of using the concept lattice structure, differences in extent and intent sets are taken into account.

[24] i.e. the user is explicitly asked to make relevance assessments in the system. Opposed to "implicit relevance feedback", where relevance assessments are "inferred" by the interaction of the user with the system.



(a) Vector-space model example. Documents $(d1, d2, d3, d4)$ and the query Q are represented as points. Axis correspond to arbitrary description spaces. Dashed lines represent the distance between the query and documents $d2$ and $d4$. The latter is closer to the query than the former meaning that it is more "relevant" for retrieval purposes

(b) Example of relevance feedback

Figure 1.6: Retrieval paradigm examples

1.4.5 Applications and Systems

Applications

Semantic retrieval. How to mix semantic technologies (what is known as semantic web) with IR techniques is still an open question. It is fair to say that modern IR systems are more focused on how to retrieve documents from very large collections than to provide reasoning or inferring capabilities to their engines. Nevertheless, this has not hindered the adoption of some of the semantic web notions such as the knowledge graph in the Google Web engine[25]. Regarding FCA-based IR approaches we can highlight the work of Messai et al. [97] presented in 2005 adapting the ideas of query refinement to support the use of ontologies for generalization purposes. In 2012, Ferré et al [55] introduced LISQL, a query language for logical information systems supporting complex relational properties among objects. These ideas were materialised in a geographical information system. Finally, in 2014 the work of Alam et al [3] presents the concept lattice as a classification of SPARQL answers to provide views on linked open data retrieval.

Recommender systems (RSs). RSs have become increasingly popular at the point that currently, it is an independent research community. Nevertheless, RSs have their roots in IR sharing many notions such as indexing, retrieval and ranking. To phrase it in the terms of [138], an important *affordance* of FCA *for* RSs is the *characterization* it can provide to recommendations, i.e. it can explain why a certain item is being recommended so the user can have a better experience with the system. This fact was addressed by [73] in 2008 which proposed a system for "well-interpretable recommendations based on FCA" for advertisement keywords using association rules. Previously, in 2006 [47] FCA was used as a method to pre-calculate groups of users that agree in certain groups of items. To achieve this the notion of *query concept* is replaced by the "entry-level concept" of a user or an item. Experimental results suggest that FCA alleviates

[25] `http://www.google.com/insidesearch/features/search/knowledge.html`

the otherwise hard task of finding the neighbourhood of a given item or user in the dataset. In 2013, Senatore et al. [130] proposed a recommender system based on an extension of FCA (namely, "Fuzzy FCA" or more precisely, FCA with fuzzy attributes) allowing to include *degrees of similarity* between users (i.e. not just Boolean relations for rating the same item) providing *ranked* recommended items. Finally, in 2014 Castellanos et al. [27] presented an approach based on [47] to extract preferences from a user activity log and derive semantically-enhanced item recommendations from them.

Others. For the sake of brevity, in what follows we provide a summarised overview of some other applications of FCA-based IR systems. **File Systems (FS)** are an interesting application in low-level information retrieval (operating system level). FCA provides a more dynamic interaction with the file system structure where the FS can be represented as a lattice instead of a tree [56, 95, 131] **Source code location** is an important task in software engineering as it enables code refactoring, among other applications [102, 119, 2]. Other interesting applications are **mathematical expression search** [109] and **multimedia indexing** [96, 50].

FCA-based IR Systems

- **FaIR (2000)** by Uta Priss [121]: A faceted IR system in which formal concepts of documents and descriptors are mapped to thesauri entries. It features a query language built on top of the set of formal concepts with the logical operators AND, OR, NOT.
- **CREDO (2004)** by Carpineto and Romano [25]: CREDO works as the front-end of a Web search engine (such as Google or Yahoo). It implements some of the authors' ideas in query expansion presented in REFINER, providing context to an otherwise plain-list of ranked documents. Extensions of CREDO included its port to mobile devices (CREDINO and SmartCREDO [18]).
- **JBrainDead (2004)** by Cigarrán et al. [31]: A FCA-based system that combines standard IR techniques such as term weighting and ranking for automated attribute selection. We highlight in this work the novel evaluation metrics of the user's effort needed to find documents w.r.t. the number of formal concepts within the lattice.
- **Mail-Sleuth and the Sleuth Family (2004 - 2009)**, Eklund, Ducrou et al.: Building on previous work, the authors present a commercial tool called Mail-Sleuth [53], a system for searching and browsing personal email collections under the assumption that novice users are able to manage a line diagram of a lattice structure. The authors extended these ideas to different application domains: ImageSleuth [50] for image browsing and retrieval (discussed in the previous sections), DVDSleuth [48] for browsing Web catalogues , SearchSleuth [49] for browsing results from a standard Web search engine and AnnotationSleuth [52] a system designed for browsing a virtual-museum collection. In 2014, Wray and Eklund presented the application *"A place for art"* [141] which followed in the steps of AnnotationSleuth with a much more elaborated user interface.
- **FooCA (2005)** by Koester [82]: In the steps of CREDO, it also relied in the assumption that users can manage line diagrams of concept lattices as well as interacting directly with the formal context.
- **BR-Explorer (2006)** by Messai et al [99]: An algorithm for document retrieval using the notions of "query concept", "pivoting" and "ranking" within concept lattices built from bioinformatic datasets.

- **Camelis (2007)** by Sebastian Ferré: Based on "a generalization of FCA" named Logical Concept Analysis (LCA) where attributes are replaced by logical formulæ. Designed to cover four main aspects: mixing query and navigation, expressive query language, genericity in data types, and efficiency for large collections. Camelis integrates several taxonomies different in nature, e.g. geographical ($Paris \sqsubseteq France$), numeric ($1999 \leq 2000$) and conceptual (ICFCA $\sqsubseteq$ Conference), allowing complex querying and other tasks previously discussed, such as "query-by-navigation" and "query-by-example". An extension of Camelis called Sewelis (or Camelis 2) was introduced in [55] for "Query-based Faceted Search" on linked data, introducing an expressive query language called LISQL (Logical Information System Query Language).
- **CreChainDo (2007)** by Nauer and Toussaint [108]: A FCA-based IR system supporting explicit relevance feedback.

1.5 Conclusions

Two related topics have been left out of the scope of this review while they remain of extreme importance for FCA-based IR approaches. Firstly, the *use of complex data* for document indexing. Several approaches have proposed more sophisticated models than the standard Boolean retrieval model defined at the beginning of this chapter. Mainly, they rely in three FCA extensions for dealing with complex data, Logical Concept Analysis such as in [57], Fuzzy FCA such as the case of [130] and Pattern Structures, such as the case of [37] or [98] (the latter does not explicitly apply pattern structures, but the notions are very similar). Secondly, the application of FCA to large collections of documents or big data (an interesting discussion is provided in [86]). Both of these matters deserve a more extensive treatment than the one we could give them here.

As we have shown through this chapter, the underlying model of FCA was one of the first considerations for a standard IR model, decades before FCA was formalized. For this reason, FCA has been extensively used for IR systems and applications. FCA-based IR approaches (as we have called them) rest over a limited pool of ideas, all of which stem from the initial Boolean IR model implementation supported over a concept lattice. These "affordances" of FCA for IR as they have been called [138], have diversified through time in a myriad of applications, ranging from usual document retrieval to semantic retrieval, mathematical expression search and file systems.

Chapter 2

Contributions on Retrieval

Contents

2.1 Prologue 33
2.2 Introduction 34
2.3 Background 35
2.3.1 Information content as a semantic relation measure 35
2.3.2 The principles of Concept Lattice-based Ranking 36
2.4 CLAIRE - Concept Lattices for Information Retrieval 40
2.4.1 Motivation for a new approach for Information Retrieval based on Formal Concept Analysis 40
2.4.2 The principles of CLAIRE 41
2.4.3 The implementation of CLAIRE as a Knowledge Discovery in Databases process 42
2.4.4 Step 1 - Document Classification 42
2.4.5 Step 2 - Concept Lattice Navigation 43
2.4.6 Step 3 - Formal Concept Ranking 46
2.5 Experimental Evaluation 46
2.5.1 Experimental setting 47
2.5.2 Evaluation measures 47
2.5.3 Results 50
2.5.4 Query analysis 51
2.6 Conclusions 53

2.1 Prologue

As described in the last chapter, one of the first models to be considered as a standard for IR systems was one that contained the main notions behind FCA, decades before it was formalized [105]. In a FCA-based IR approach, we use a concept lattice as a document index where the search space is given by the formal concept intents and their order. Our research starts from this point. In this chapter we present our work which focuses on how to use a document index modelled as a concept lattice for retrieval purposes. Our goal is two-fold. Firstly, we want to test whether the lattice structure can be exploited further than state-of-the-art techniques

to support a semantically-enhanced automatic retrieval approach. Secondly, we want to understand the benefits and drawbacks of the concept lattice as a document index. Lessons from this experience will be further explored in the next chapter.

2.2 Introduction

The increasing amount of information available nowadays implies more and more the ability to accurately retrieve documents relevant to user needs. Regarding this task, several approaches have been proposed in the field of information retrieval (IR) [93]. Document retrieval for example, a sub-task of IR, focuses on how to exploit the basic elements of information that documents contain (terms, phrases, links, etc.) and their in-between relations, in order to construct a document index that users can browse and query. However, as document descriptions become more complex, high-dimensional and domain-specific, these information elements become too limited in their capacity to identify relevant documents for a given user information need, and thus other factors, such as semantics, need to be considered for the purpose of document indexing and retrieval. Consequently, semantic indexing for document retrieval has gained importance in the IR literature [51]. In this chapter we present our approach for document retrieval, a novel technique to combine relations among documents through the terms they share, and the semantics of those terms. To achieve this, we combine two typical document retrieval techniques, namely *"navigation"* among document relations and *"ranking"* of documents, using a notion of similarity between the semantics of document terms and the keywords of a user query. Both techniques, as shown in [24], can be naturally modelled and implemented in a document-term concept lattice computed from a document collection.

More formally, our model relies on the general idea of constructing a document-term concept lattice used as an index to answer a given user query. The benefits of using a concept lattice as a query index are two-fold. Firstly, the concept lattice provides a structured support for the full query space (possible queries in a document collection), since it contains all possible modifications (or variations) of the original user query and their corresponding documents, organized in a partial order. Therefore, it allows considering the problem of document retrieval as a problem of navigation inside the lattice, starting from an original "query concept" and following the principles of classification-based reasoning [106]. Secondly, the lattice allows an easy incorporation of domain knowledge at attribute (term) level, thus significantly improving the semantic aspect of document retrieval that we need to address. For this, we develop a novel concept lattice exploration strategy based on the notion of "cousin concepts", as well as a new approach for ranking concepts based on their in-between semantic similarities [59], measured using an external lexical hierarchy. In the same way we anchor our document indexing and retrieval technique to the formal concept analysis definitions, we frame the entire process (comprising from document analysis to results presentation) within the knowledge discovery in databases (KDD) framework [16]. From a process design perspective, we take advantage of the robust and clear KDD process to guide the main steps to be completed in order to create an IR system which satisfies the users' information needs, in this case represented by small sub-sets of documents of vast document corpora. From a theoretical perspective, KDD is an accepted framework with a well established supporting community and an extensive literature regarding its relations with FCA [117].

The main contributions of our approach are the introduction of cousin concepts and the use of semantic relations to enable document ranking on a concept lattice-based information retrieval system, built and described as a KDD-like process. Finally, we validate our model using 4 typical IR document datasets and comparing it to 3 currently used document retrieval

techniques. Results show that our approach achieves better performance for most of the ranking evaluation methods used.

2.3 Background

2.3.1 Information content as a semantic relation measure

We can measure the semantic relation between any two terms by using the measure known as *Lin similarity* [89]. This is a measure related to the amount of information carried by a term or a word within a context (piece of text, document or corpus). Consider for example that we have a medical document d_1 indexed by two terms: arthroscopy[26] and complication. In this scenario, our job is to find similar documents to d_1. In order to do so, we take one of the terms within d_1 (let us pick the term arthroscopy) and look for other documents that contain that term. Consider that we have two of such documents, the first containing terms arthroscopy and practice, and the second containing arthroscopy and infection. The question now is how to identify which of these documents is more similar to the original one. Intuitively, we may choose the one with the term infection which supposes a kind of "complication" in the context of a surgery such as an arthroscopy (indeed, a very serious complication), while the term practice is much more general, making the document with that term less similar to the original. This notion of information correlation between two terms, or between the information content shared by them in a given context, can be measured with Lin similarity which takes in consideration the actual frequency correlation in a text corpus as well as the commonalities those terms have in a lexical hierarchy (such as a dictionary). To formalize, given two terms m_1 and m_2, the Lin similarity between m_1 and m_2 is defined as:

$$lin(m_1, m_2) = \frac{2 \log p(m_s)}{\log p(m_1) + \log p(m_2)} \tag{2.1}$$

Where $p(m_i)$ is the probability of the term m_i to appear in a given document of the corpus and m_s is the "lowest common subsumer" of terms m_1 and m_2 in the lexical hierarchy. In this work, we use the *Brown corpus*[27], as the document corpus to measure the probability of term appearance, and *WordNet* as the lexical hierarchy that will yield the lowest common subsumer of the terms. These resources were selected since they are widely used in IR systems. Nevertheless, our approach is not restricted to use them exclusively and they can be replaced by other similar resources related to a given domain. In the following, we provide a short description for these resources.

Brown Corpus. The Brown Corpus is a general text collection, which contains samples of 500 English language text documents, and approximately one million words, widely used in text linguistics. The Brown corpus was used in this work to calculate term frequencies ($p(m_i)$ in Equation 2.1).

WordNet. WordNet is a well-known semantic dictionary, which associates terms with their meanings, called synsets (as "set of synonyms") [103]. Each term in WordNet may be associated with several synsets, where each synset corresponds to one specific meaning of the term. Synsets inside WordNet are organized in a hierarchical tree structure, based on their hypernym/hyponym relations. In this work we use Wordnet to obtain the lowest common subsumer m_s in Equation 2.1. The lowest common subsumer for two synsets is simply the most general synset in the Wordnet hierarchy which is a hypernym for them both.

[26] Arthroscopy refers to a surgery on a joint using an arthroscope.

[27] `http://khnt.aksis.uib.no/icame/manuals/brown/`

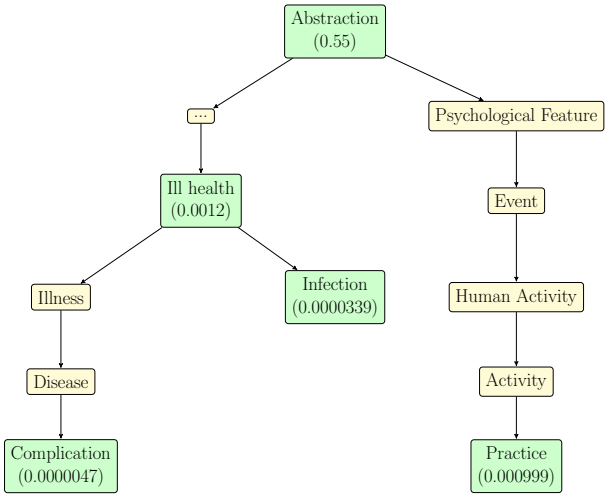


Figure 2.1: Terms complication, infection and practice as positioned inside the lexical hierarchy Wordnet, shown in darker boxes together with their minimal common subsumers. The box with three dots represents 5 unimportant terms in the hierarchy.

Let us see an example of using both external knowledge sources and the Lin similarity measure, to calculate semantic similarity between terms. In Figure 2.1, terms complication, infection and practice are shown along with their lower common subsumers ill health and abstraction (darken boxes), as found in WordNet. The number under each term is the probability of appearance of that term in the Brown corpus. It may observed that the term complication is very close to the term infection (the sense of complication defined as: "any disease or disorder that occurs during the course of (or because of) another disease"). On the other hand, practice is very far from the term complication (14 steps in the tree compared to only 4 for infection) sharing the lower common subsumer abstraction (defined as "a general concept formed by extracting common features from specific examples"). This is confirmed by the Lin similarity value of the two candidate term pairs: 0.59 for the complication-infection pair, and only 0.062 for the complication-practice pair, which leads to the selection of infection as the term to replace complication.

2.3.2 The principles of Concept Lattice-based Ranking

Section 1.4.4 presents the state-of-the-art w.r.t. to FCA-based IR approaches focusing in the automated retrieval of documents using ranking (i.e. not based on relevance feedback). In this section we provide further details on the technical aspects behind these works. In what follows, we will re-use the example provided in Section 1.4.1.

Consider the formal context in Table 1.8 and the user query $\mathtt{q_i}$ containing terms arthroscopy and complication (hereafter we refer to the terms within a query as *keywords*). As discussed in Section 1.4.4, the typical FCA-based IR model considers the representation of the query as a "virtual formal concept" within the concept lattice. For this purpose, let us define a *virtual object* that can be included in the formal context as any other object. Thus, the original formal context is redefined to include the query $\mathtt{q} = (\mathtt{q_e}, \mathtt{q_i})$, where $\mathtt{q_e}$ is the virtual object and $\mathtt{q_i} \subseteq \mathtt{M}$ contains its keywords (i.e. the constraints associated to the query). The new formal context is denoted as $\mathcal{K}_\mathtt{q} = (\mathtt{G} \cup \{\mathtt{q_e}\}, \mathtt{M}, \mathtt{I} \cup \{(\mathtt{q_e}, \mathtt{m})_{\forall \mathtt{m} \in \mathtt{q_i}}\})$ and its associated concept lattice is computed using a FCA algorithm.

The concept lattice computed for the formal context of Table 1.8 (including the query) is

illustrated in Figure 2.2[28]. After constructing the lattice, the standard procedure in the CLR family approaches is to find the *object concept* of the virtual object $\mathsf{q_e}$. This concept is usually called the *query concept* and it is the starting point to find documents satisfying a query in the lattice [97, 100, 24].

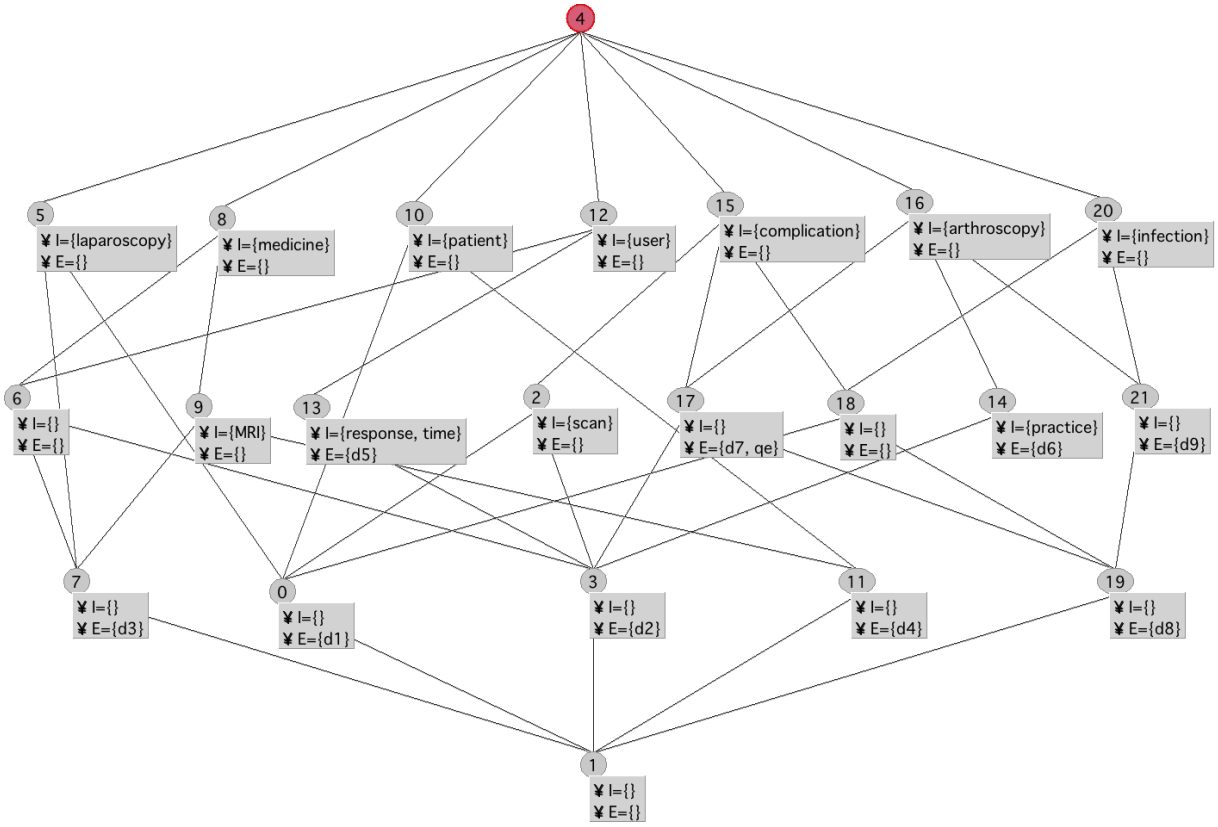


Figure 2.2: Concept lattice in reduced notation derived from a document-term formal context including the query. The reduced notation of a lattice consists in labelling the extents/intents only with the first appearance of an object/attribute from top-to-bottom/bottom-to-top (respectively), i.e. objects are show in their *object concepts* and attributes in their *attribute concepts* (e.g. concept 19 is the *object concept* of document d_8 and concept 15 is the *attribute concept* of the attribute complication).

Let us continue with the above example. The *query concept* for the query with keywords $\mathsf{q_i}$ is concept 17 in the concept lattice illustrated in Figure 2.2. Its intent contains terms arthroscopy and complication. Its extent contains the *virtual object* $\mathsf{q_e}$ and documents d_2, d_7 and d_8 which satisfy a *conjunctive* version of query $\mathsf{q_i}$, i.e. these documents include *all* the query keywords. We refer to these documents as the *exact answer*.

However, very often documents relevant to the user may fail to meet the restrictions of a *conjunctive query* due to different issues, such as language ambiguity (e.g. synonymity or polysemy of terms), poor document descriptions, the lack of user's knowledge about how to effectively pose a question or create a query, etc. This is known as the *non-matching document problem* [24], which refers to the fact that documents relevant to the user query may not always exactly match its keywords and therefore they are not included in the exact answer. To overcome

[28]Figure drawn using Galicia software - `http://www.iro.umontreal.ca/~galicia/`

this issue, it is possible to use the lattice to satisfy *disjunctive* versions of the query, i.e. retrieve documents that contain only some query terms, by using the super-concepts of the *query concept*. For this example, concept 16, a super-concept of 17, contains in its extent document d_6 and in its intent the term arthroscopy, while concept 15, also a super-concept of 17, contains in its extent document d_1 and in its intent the term complication. We say that these documents "partially" meet the query and they provide a *close* or *partial answer*.

As it can be observed in Figure 2.2, each formal concept in the concept lattice contains a possible conjunctive query and a set of documents which satisfy that query, while combinations of formal concepts (in the form of unions) work analogously for disjunctive queries. In fact, the concept lattice configures the *global query space* of the document collection, where the query concept represents the original user query and its super and sub-concepts represent the immediate modifications that can be performed over that query to find "partial-matching documents". Notice that only super-concepts contain different documents than those that could be found on the *query concept*, since sub-concept's extents will always be subsets of the query concept's extent. Following the idea of disjunctive queries, aside from the query concept and its super and sub-concepts, different formal concepts within the lattice may contain intents that have a non-empty intersection with the query and thus, they can be also used to generate query modifications. Finally, some formal concepts within the lattice do not share terms with the user query and do not constitute query modifications. It is important to notice that, given that the lattice forms the global query space of the document collection, the retrieval of those concepts that represent meaningful query modifications can be considered as a matter of: 1) *navigating* the lattice starting from the query concept and then 2) *ranking* the retrieved concepts w.r.t. their relations with the query concept.

Current lattice navigation and ranking approaches

Two main different navigation strategies have been proposed in the literature. The neighbourhood expansion strategy [24] is based on the idea of visiting concepts, in an "expanding ring" order, starting from the query concept. This strategy does not make a distinction between visiting super or sub-concepts, since in the same ring there may be super and sub-concepts of the query concept. The hierarchical exploration strategy [100] navigates the lattice by exploring the super-concepts of the query concept. These super-concepts contain more documents than those found in the query concept thus allowing to work with a disjunctive approach.

Both strategies assume a topological distance measure in order to *rank* the concepts reached by *navigation*. In this work, the topological distance in a lattice is defined as the minimal path length between two given formal concepts (considered as nodes in a graph [132], see also the nearest neighbour relation in [24]). This notion is straightforward in the sense that nearer concepts from the query concept are considered "more related" and hence, they receive a better ranking. However, both strategies differ in that using the neighbourhood expansion it is possible to reach many more concepts within the lattice than in the case of hierarchical exploration.

Regarding *query modification*, the hierarchical exploration strategy works by modifying the original conjunctive query to a set of disjunctive queries represented by the intents of the super-concepts of the query concept. From these super-concepts it is possible to obtain a set of documents used as an answer for the original query. For example, Figure 2.3 presents the section of a lattice containing 4 concepts including a query concept (concept 3 in white) for the conjunctive query arthroscopy *AND* complication *AND* practice (notice that in this case the marker q_e is on concept 3) using the hierarchical exploration strategy. Both super-concepts of the query concept (concept 14 and 17) receive the same ranking (they are both at distance

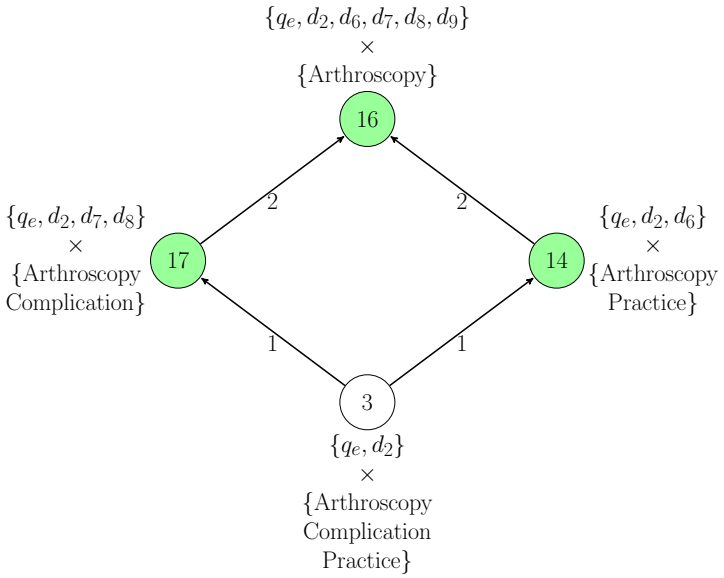


Figure 2.3: Section of a lattice showing 4 concepts obtained by hierarchical exploration. Arrows represent the navigation direction with their correspondent topological distance from the query concept of query arthroscopy, complication and practice (represented in white).

1 from the query concept) and hence the answer is the union of the extents of both super-concepts. Actually, this is the answer for the disjunctive query (arthroscopy *AND* complication) *OR* (arthroscopy *AND* practice) or more shortly, arthroscopy *AND* (complication *OR* practice). In this manner, hierarchical exploration searches in the query space for *relaxed* versions of the original query and rank them according to a notion of *relaxation*, i.e. the more relaxed the query, the lowest ranking it has (notice that the concept 16 ranked at distance 2 answers the very relaxed query containing only the keyword arthroscopy).

The notion of query modification is not explicitly present in the neighbourhood expansion strategy since in the same "ring" of concepts (i.e. those at the same distance from the query concept) different types of concepts are considered and ranked equally. For example, in Figure 2.4 the same conjunctive query arthroscopy *AND* complication *AND* practice is represented along with 6 other concepts obtained through neighbourhood expansion. There are 4 "rings" represented by the topological distances included in the arcs between concepts (e.g. ring 1 contains concepts 14 and 17). It can be appreciated that concepts with different intent cardinalities receive the same ranking since they are in the same "ring" (e.g. concepts 16 and 19 in the ring 2). Moreover, it is difficult to assess the modification in the query represented in ring 4 (concept 20) containing the keyword infection. Nevertheless, this characteristic also gives neighbourhood expansion its potential since it is able to find many more documents than the hierarchical exploration strategy (for example, with hierarchical exploration the concept with the term infection is not a possible query modification and document d_1 is never considered as an answer). As such, there is not an actual notion of query modification, but an idea that closer concepts in the concept lattice will contain closer document descriptions and hence, closer relevant documents.

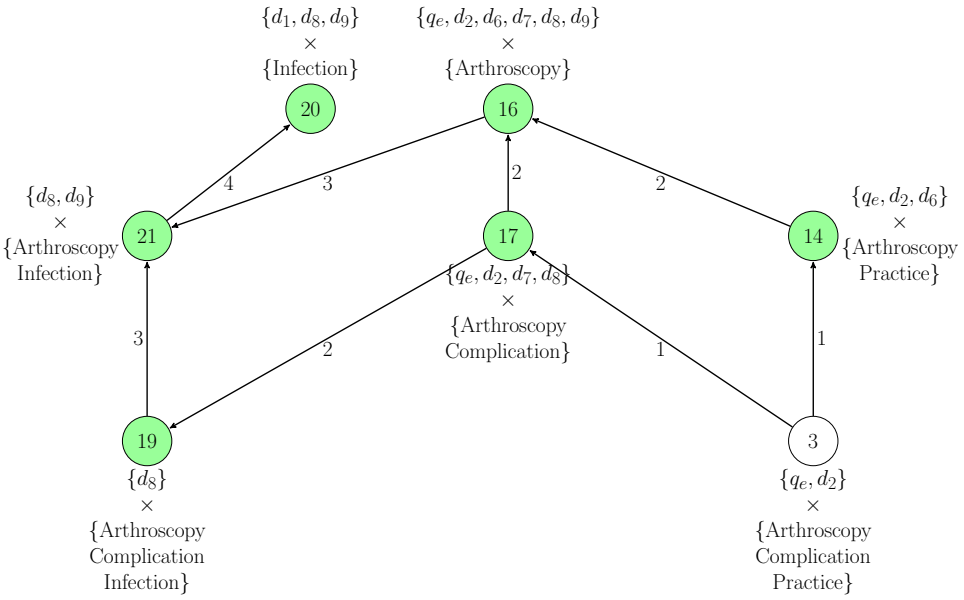


Figure 2.4: Section of a lattice showing 7 concepts obtained by neighbourhood expansion. Arrows represent the navigation direction with their correspondent topological distance from the query concept of query arthroscopy, complication and practice (represented in white).

2.4 CLAIRE - Concept Lattices for Information Retrieval

2.4.1 Motivation for a new approach for Information Retrieval based on Formal Concept Analysis

As described in the previous section, the main difference between hierarchical exploration (HE) and neighbourhood expansion (NE) strategies is how the notion of *query modification* is applied. Since HE is based on a clear *query relaxation* process where documents are ranked according to how relaxed is the query they satisfy (w.r.t. the original query), we can expect that the answers that HE provides, compared to those obtained from NE, are of better quality in terms of relevant documents. On the other hand, since NE is based on a continuous expansion of the lattice region used to retrieve documents, we can expect that the answers it provides contains a larger quantity of relevant documents compared to the answers provided by HE (along with a larger quantity of irrelevant documents).

The trade-off between quality and quantity has always been an active issue in the IR domain, mainly reflected by the two most common retrieval evaluation measures: *precision* and *recall* [93] (they will be detailed in Section 2.5.2). Furthermore, it is hard to compare a system with high quality in the answers versus one with high quantity of answers since, in many cases, this depends on the application intended for the system. For example, while looking for a restaurant in an unfamiliar city, the user may be interested in looking through a list of all those available (focus on quantity). Instead, while looking for a restaurant in his own city, the user may be interested in a specific type, brand or locality (focus on quality of the answer). Nevertheless, it is accepted that a good retrieval system should have a good quality/quantity balance which is our goal in this work. To achieve this, we take lessons from HE and NE strategies considering a careful design in the evaluation of the *query modification* process and (*ranking*), but also considering an expansion to formal concepts within the lattice, other than those in the super-hierarchy of

the query concept (*navigation*). These two elements are reflected by two of the three aspects of our work, namely classification, navigation and ranking. Regarding navigation, we define a new relation for two given concepts within the lattice which we call *cousin concepts*. Regarding ranking, we consider a semantic-based formal concept similarity measure introduced in [59]. In the following, we describe and detail the three main aspects of our approach called CLAIRE (Concept Lattices for Information REtrieval).

2.4.2 The principles of CLAIRE

CLAIRE focuses on the following three aspects.

1. *Classification:* In this work, classification is used with two meanings, namely the *operation* of classification and the *product* of this operation which is also called "a classification". Firstly, we use FCA for building a concept lattice which is considered as a semantic index for document retrieval (the concept lattice as a result of a classification operation). Then, given a user query, we rely on the principle of *classification-based reasoning* for inserting the query in the lattice and identifying formal concepts that provide possible answers to the query (the classification operation applied to the query). This method of query insertion differs from those used in the CLR family approaches. In CLAIRE, as detailed previously, the query concept is not appended to the lattice through an incremental FCA algorithm (e.g. Galois in the case of CLR [19]), but it is classified by the lattice through classification-based reasoning.

2. *Navigation:* We propose a new navigation strategy of the concept lattice which is tailored to the needs of the ranking method proposed using a semantic similarity measure. While we propose a navigation to be used in the same sense as in CLR-like approaches, i.e. the identification of relevant concepts given an initial query concept, our proposition is based on the notion of *cousin concepts*. The rationale behind the use of cousin concepts is that in order to identify additional, partial-matching documents we need to modify the original user query but in a manner that the query and the modified query are closely related. We achieve this by the generalization of the query concept in the concept lattice to its super-concepts (which we call *query generators*) and their posterior specialization to what we call *cousin concepts* of the query concept (i.e. the sub-concepts of the query generators). Since query generators are immediate super-concepts of the query concept, cousin concepts retain some keywords (more precisely, those in the query generators) while including some other terms. For example, consider the scenario where a query concept has the intent $\{\mathtt{m}_1, \mathtt{m}_2\}$ and its query generator with intent $\{\mathtt{m}_1\}$. Through the specialization of the query generator we could obtain the concepts with intents $\{\mathtt{m}_1, \mathtt{m}_3\}$ and $\{\mathtt{m}_1, \mathtt{m}_4\}$ which are considered as modifications of the original query (i.e. replacing $\mathtt{m}_2$ with $\mathtt{m}_3$ or $\mathtt{m}_4$, respectively).

3. *Ranking:* Since many cousin concepts (or query modifications) can be obtained from the concept lattice for a single query concept, there is a need to evaluate how *close* are these modifications from the original query. For the previous example the question is whether we should replace $\mathtt{m}_2$ with $\mathtt{m}_3$ or with $\mathtt{m}_4$. We answer this by measuring the *semantic similarity* of the terms included in the intent of each retrieved concept w.r.t. the keywords. In this way, we also address the problem of retrieving documents related to the user query in a semantical way, rather than only based on topological characteristics of the concept lattice. We use a measure introduced in [59] which considers external knowledge sources to evaluate "semantic closeness".

2.4.3 The implementation of CLAIRE as a Knowledge Discovery in Databases process

Following the rationale described above, here we describe the proposed CLAIRE approach for document retrieval, which considers *classification*, *navigation* and *ranking* based on the notions of *classification-based reasoning*, *query modification* and *semantic similarity*, respectively. We formulate our approach following the lines of a knowledge discovery in databases (KDD) process [16] allowing well differentiated tasks within a robust framework which implements the document retrieval process (Figure 2.5). In particular, our approach is defined as a sequential three steps KDD-like process to reflect the three aspects of our work (*classification*, *navigation* and *ranking*). The first step of our approach is "Document Classification", related to the data pre-filtering step of a KDD process. The second step is "Lattice Navigation" related to the mining/knowledge discovery KDD-process step. Finally, the last step of our process is "Concept Ranking", related to the interpretation step.

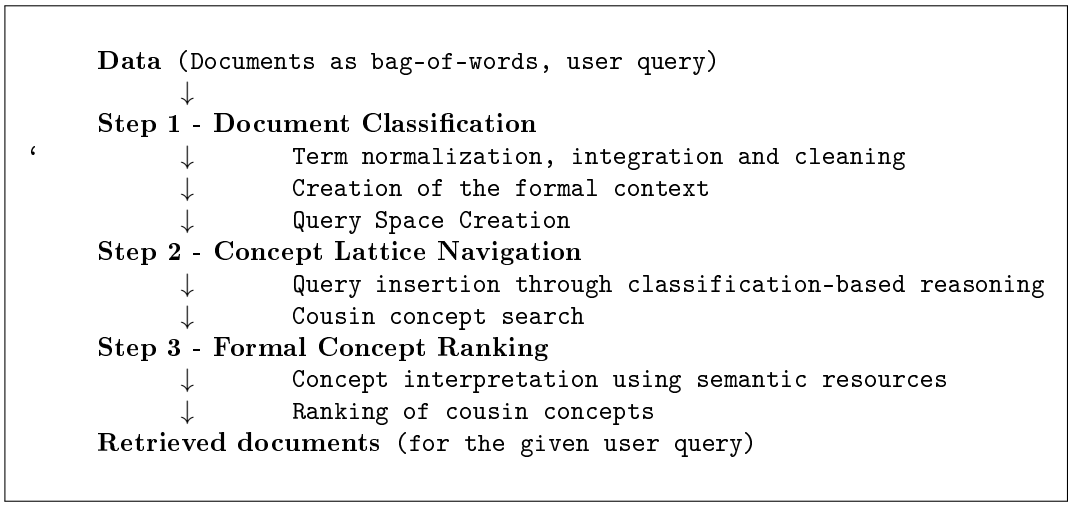


Figure 2.5: 3-step KDD-like document retrieval process.

2.4.4 Step 1 - Document Classification

In this step we construct the formal context $\mathcal{K} = (\mathtt{G}, \mathtt{M}, \mathtt{I})$ as in traditional FCA-based IR approaches. Depending on the nature of the collection of documents, different tasks should be performed in order to construct the formal context (e.g. parsing, tokenizing, stop-word filtering etc. [94]). In order to simplify and standardize the approach, we assume that the documents in the collection are in the form of a bag-of-words (i.e. each document consists of a set of terms). We argue that this is a safe assumption since most of document corpora are already provided in this format and in the other case, the transformation of text to bag-of-words is a straightforward process. Additionally, a normalization of the terms is required in order to reduce sparsity and integrate the representation of documents. Three basic natural language processing techniques can be used [94]. *Stemming* is a technique that normalizes a set of words to their morphological root (e.g retrieval, retrieves, retrieve are normalized to retriev). Thus, it greatly reduces the sparsity and the number of attributes in the context, however it does not maintain the original meaning of the terms (e.g. retriev is not an actual word). To maintain the meaning

of the terms it is possible to use a *semantic element mapping* which normalizes a set of words to a semantic element definition using an external knowledge source (e.g. recover, retrieve, find, regain are mapped to the definition *"Get or find back; recover the use of"*)[29]. This technique reduces sparsity but produces an explosion in the number of attributes in the context, since each term can be mapped to more than one definition (e.g. recover maps to 4 different definitions). Finally, through the use of *lemmatization* it is possible to normalize terms to their inflected forms (e.g. retrieval, retrieves, retrieve are normalized to retrieve). This technique slightly reduces the sparsity of annotations and the number of attributes in the context, while maintaining the original meaning of the terms.

Given the complexity of calculating a concept lattice and the fact that we need to maintain the original meaning of the terms in order to measure their in-between semantic similarity, we normalize document terms using the technique of word lemmatization.

Finally, a concept lattice representing the query space is created based on the document-term formal context associated to the document corpus. Different algorithms exist to compute a set of formal concepts from a formal context and to build a concept lattice [87]. For this work, we rely on the AddIntent algorithm [139] given its performance for computing formal concepts along with their order, i.e. the concept lattice

2.4.5 Step 2 - Concept Lattice Navigation

The second step corresponds to navigating the constructed "document index" or "query space" in order to retrieve documents for a given user query[30]. For convenience, we propose a model-based document retrieval approach, i.e. the document index is built a-priori and not for each given user query like in usual CLR-like approaches (described in Section 2.3.2). This is done given the complexity of building a concept lattice from a formal context of significant size. Instead, in our approach the concept lattice is constructed once and the query is simply "inserted" (actually, classified) using the concept lattice when required.

Query insertion through classification-based reasoning

This sub-step assumes the existence of the concept lattice and a user query in the form of a set of keywords. Its output is a query concept and a set of related formal concepts which are used to retrieve a set of documents. A user query $\mathtt{q}$ is considered as a *query concept* $\mathtt{q} = (\mathtt{q_e}, \mathtt{q_i})$ where $\mathtt{q_e}$ is a "dummy variable" to be instantiated by retrieved documents and $\mathtt{q_i} \subseteq \mathtt{M}$ is a set of keywords.

The query concept is not actually "inserted" in the lattice, but rather "classified" by it using *classification-based reasoning* as introduced in [106]. Classification-based reasoning is based on a depth-first traversal of the lattice and consists in, given a query concept $\mathtt{q}$, searching for the *most specific subsumers (MSS)* and the *most general subsumees (MGS)* of $\mathtt{q}$. Actually, the search for the most general subsumees here is useless. The search for the most specific subsumers is illustrated in the algorithm of Figure 2.6 and works as follows. The classification-based reasoning algorithm receives a concept to classify $\mathtt{q}$ and a MSS candidate concept $\mathtt{C}$ (line 1). $\mathtt{C}$ is firstly checked for if it was previously visited and if not, then it is marked (lines 2-4). This is done to ensure that concepts are visited only once. A subsumption test is performed for checking whether

[29] Dictionary definition of the first sense of recover given by Wordnet.

[30] Notice that "document index" or "query space" are both a dual view of the same concept lattice from the point of view of extents or intents, respectively. Hereafter we refer to "document index", "query space", "concept lattice" or "semantic index" indistinctly.

```
1 function compare (q, C)
2    if C is marked
3      then return {}
4    else mark C
5    if C does not subsume q
6       then return {}
7    else MSS ← {}
8      for every descendant E of C do
9        MSS ← MSS ∪ compare (q, E)
10   if MSS = {}
11      then return {C}
12   else return MSS
```

Figure 2.6: Classification-based Reasoning algorithm: Searching for the most specific subsumers of $\mathtt{q}$: comparison of the current object $\mathtt{C}$ with $\mathtt{q}$.

$\mathtt{C}$ subsumes $\mathtt{q}$, where the subsumption test relies on intent inclusion: $\mathtt{C}$ subsumes $\mathtt{q}$ as long as the intent of $\mathtt{C}$ is included in the intent on $\mathtt{q}$ (line 5). If $\mathtt{C}$ does not subsume $\mathtt{q}$, the sub-lattice rooted in $\mathtt{C}$ is cut and no longer considered (line 6). If $\mathtt{C}$ subsumes $\mathtt{q}$, a recursive call of the classification-based reasoning algorithm is called over the sub-lattice rooted in $\mathtt{C}$ (line 7-9). The traversal continues with the first descendant of $\mathtt{C}$ and so on in the same way. The traversal ends when there are no more concepts to visit and returns the set MSS of most specific subsumers. In the case that no MSS were found in the sub-lattice of $\mathtt{C}$, then $\mathtt{C}$ becomes a MSS (lines 10-12) In the present case, these most specific subsumers are called *query generators*.

In case there exists in the lattice a formal concept $(\mathtt{A}, \mathtt{B})$ such as $\mathtt{B} \equiv \mathtt{q_i}$, then the algorithm identifies $(\mathtt{A}, \mathtt{B})$ as the query concept, and those documents in $\mathtt{A}$ constitute the *exact answer*. The existence of an *exact answer* is not always guaranteed, especially for large and complex queries. The worst case scenario appears when no *query generators* are found except for the top concept of the lattice (which includes all documents and no terms[31]). Actually, this can only be the case if no keyword provided by the user can be found in the query space and in this case, the query is considered to be *unsuccessful*.

As we have previously described, for a user query represented by a query concept, query generators represent "relaxed versions" of the original user query, i.e. they include fewer keywords than the query intent. Any other sub-concept of a query generator –except the *query concept*– induces a modified user query, since it includes a part of the query determined by the query generator, plus a set of terms that are not contained in the original query. Based on this observation, we introduce a *navigation strategy* which allows finding "successful modifications" of an original user query, i.e. they do include answers, which are closely related one to the other. In order to find and reuse these query modifications, we define hereafter the notion of *cousin concepts*.

Cousin Concept Definition:

Two formal concepts $(\mathtt{A}_1, \mathtt{B}_1)$ and $(\mathtt{A}_2, \mathtt{B}_2)$ which are not comparable for $\leq_{\mathcal{K}}$ are said to be *cousin concepts* iff there exists $(\mathtt{A}_3, \mathtt{B}_3) \neq \top$ such that:

[31] If the top concept contains a term in its intent, it would imply that all documents share that term. If such a case, that term would not help in the search of documents and can be removed from the formal context.

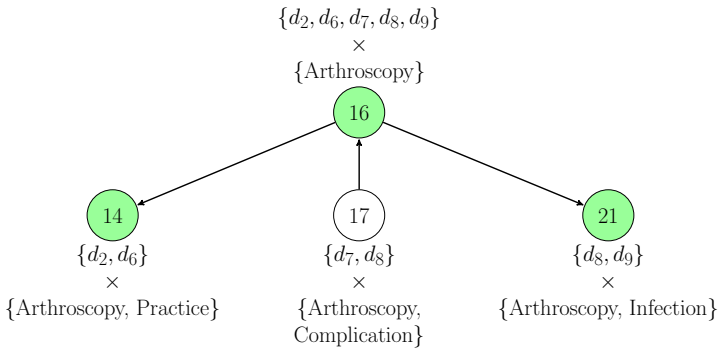


Figure 2.7: Example of lattice navigation. Starting from the query concept (17) we navigate to its cousin concepts (14,21) and retrieve documents d_2, d_6 and d_9 (document d_8 is already in the query concept and it is not retrieved again). Concepts are shown with their extents and intents. Arrows show the direction of the navigation inside the lattice.

- $(\mathtt{A}_1, \mathtt{B}_1) \leq_{\mathcal{K}} (\mathtt{A}_3, \mathtt{B}_3)$.
- $(\mathtt{A}_2, \mathtt{B}_2) \leq_{\mathcal{K}} (\mathtt{A}_3, \mathtt{B}_3)$.
- $\mathcal{D}_{\mathcal{K}}((\mathtt{A}_2, \mathtt{B}_2), (\mathtt{A}_3, \mathtt{B}_3)) = 1$ and $\mathcal{D}_{\mathcal{K}}((\mathtt{A}_1, \mathtt{B}_1), (\mathtt{A}_3, \mathtt{B}_3)) = 1$.

Where $\top$ is the top concept and $\mathcal{D}_{\mathcal{K}}$ measures the "topological distance" between two formal concepts in the lattice. The distance $\mathcal{D}_{\mathcal{K}}$ is analogous to the "minimal path length" for two nodes in a graph as defined in [132]. Intuitively, this means that $(\mathtt{A}_1, \mathtt{B}_1)$ and $(\mathtt{A}_2, \mathtt{B}_2)$ do not subsume each other and that $(\mathtt{A}_3, \mathtt{B}_3)$ is the upper bound $(\mathtt{A}_1, \mathtt{B}_1) \vee (\mathtt{A}_2, \mathtt{B}_2)$. Actually, $(\mathtt{A}_3, \mathtt{B}_3)$ represents a *query generator* of queries $(\mathtt{A}_1, \mathtt{B}_1)$ and $(\mathtt{A}_2, \mathtt{B}_2)$. This also means that the query in $(\mathtt{A}_1, \mathtt{B}_1)$ is considered as a modification of the query in $(\mathtt{A}_2, \mathtt{B}_2)$ and vice versa. We restrict *query generators* not to be the top concept, since the empty query can be considered as the generator of the whole *query space*. For example, in Figure 2.2, concept 18 is a cousin of 17 because of concept 15, concept 6 is a cousin of 13 because of concept 12 and so on. However, concept 10 is not a cousin concept of 12 since that would mean that the query generator should be concept 4 which is the top.

For a given query and its *query concept*, the querying process aims at traversing the lattice to extract all its *cousin concepts* $(\mathtt{A}_i, \mathtt{B}_i)$. For simplicity reasons we have restricted the cousin concepts to be at a topological distance 2 from each other (i.e. one up, one down). This restriction can be relaxed in order to increase the number of documents retrieved by the process if necessary.

As an example of the *lattice navigation* strategy described above, consider Figure 2.7, which contains part of the lattice in Figure 2.2. Specifically, Figure 2.7 displays concepts 14, 16, 17 and 21, where concept 17 (in white) represents the *query concept* for the query with keywords arthroscopy and complication. Its extent contains the *exact answer*, i.e. documents d_7 and d_8. Concept 8 contains in its intent just the term arthroscopy and provides a relaxed version of the original query working as a *query generator*. From this concept we can obtain the cousin concepts of concept 17, i.e. concepts 14 and 21. These provide two different query modifications where the term complication in the original query is replaced with terms practice or infection, allowing choosing whether document d_2 and d_6 or document d_9 should be ranked first. The decision on the ranking of the retrieved concepts, and the documents that they include is a matter of *interpretation* of the results and it is described in the following step.

2.4.6 Step 3 - Formal Concept Ranking

Given the query concept and its *cousin concepts*, the output of the concept ranking step is a sorted list of documents retrieved to the user. As we recall from the previous step, a cousin concept represents both a query modification (in its intent) and a set of documents that satisfy that modification (its extent). In this step we interpret these query modifications in the sense of semantic similarity w.r.t. the original user query, considering that those modifications which deviates less from the original query (and hence, are more similar to the query) should yield documents more relevant. To achieve this, we use a semantic similarity measure defined for two formal concepts within a concept lattice.

Computing the similarity between concepts

In our framework, the ranking of the candidate concepts is performed using the semantic similarity metric proposed by Formica [59]. Given two formal concepts $\mathtt{C}_1 = (\mathtt{A}_1, \mathtt{B}_1)$ and $\mathtt{C}_2 = (\mathtt{A}_2, \mathtt{B}_2)$ the similarity between $\mathtt{C}_1$ and $\mathtt{C}_2$ is defined as follows:

$$sim(\mathtt{C}_1, \mathtt{C}_2) = \frac{|\mathtt{A}_1 \cap \mathtt{A}_2|}{max(|\mathtt{A}_1|, |\mathtt{A}_2|)} * w + \frac{\mathcal{M}(\mathtt{B}_1, \mathtt{B}_2)}{max(|\mathtt{B}_1|, |\mathtt{B}_2|)} * (1 - w) \tag{2.2}$$

Where $0 \leq w \leq 1$ is a weighting parameter and $\mathcal{M}(\mathtt{B}_1, \mathtt{B}_2)$ is the maximization of the sum of the *information content* similarities between each possible pair of terms created using one term from $\mathtt{B}_1$ and another from $\mathtt{B}_2$. *Information content* similarity between two terms is measured using their distance in a lexical hierarchy and/or their co-occurrence in a text corpus (see Section 2.3.1).

As an example of how concept ranking is performed using Formica's similarity, let us consider Figure 2.7. Given the query represented by concept 17, we navigate the lattice and find two cousin concepts, in this case concept 14 (containing documents d_2 and d_6) and 21 (containing document d_9). In order to decide which of these concepts yields the most relevant documents we compare their similarity to the query concept using Formica's metric defined in Equation 2.1 with $w = 0.5$. In addition, we will use Wordnet as the external lexical hierarchy to compare terms and the Brown corpus as the body of text to locate term frequencies. We observe that $sim(17, 21) = 0.6137$, while $sim(17, 14) = 0.225$, because the pair (complication, infection) has a higher semantic relation than the pair (complication, practice), as explained in Section 2.3.1, and the intersection between the extents of concepts 17 and 14 is empty, while for 17 and 21, the extent intersection contains one element (d_8). Therefore, we may retrieve document d_9 before documents d_6 and d_2 (thus, ranking it first). Different weight values w allows for a parametrization in the preference between the structural (from the extents) and the semantic (from the intents) similarities of the compared concepts.

2.5 Experimental Evaluation

To test the capabilities of our approach, we applied it on four datasets of the SMART collection[32] which is a well known benchmark used in text mining and retrieval. Each dataset is provided as a collection of documents from different domains. Additionally, a list of queries (in different formats) is given for each dataset. A query has an associated set of valid answers, i.e. documents that have been labelled by human experts to be relevant for that query.

[32] ftp://ftp.cs.cornell.edu/pub/smart/

2.5.1 Experimental setting

All datasets are preprocessed by parsing, stop word removal and lemmatization using the natural language toolkit (NLTK) library[33] for Python. Table 2.1 details each dataset in terms of the number of document, terms, annotations (the number of document-term relations), formal context density (#annotations / (#documents × #terms)) and the number of queries with provided answers used in the experiments.

For each dataset, a formal context containing all documents and lemmatized terms is created and a concept lattice is derived from it using an implementation of the addIntent algorithm [139]. Preprocessed datasets are stored in a relational database for further operations. Concept lattices were modelled as directed graphs using the networkX library[34] for handling large graphs. Each received query is processed to construct a bag-of-words using its lemmatized keywords. A query concept is created including all the lemmatized keywords in its intent and an empty extent. Using classification-based reasoning we look for the *query generators* of the query concept. To compute Formica's similarity, the query concept extent is considered as including the *union of the extents of all its query generators*. This heuristic greatly improves the performance of the posterior ranking in the four datasets used. We provide a further explanation in the following discussion. The sub-concepts of the *query generators* (cousin concepts) are ranked using Formica's similarity measure described in Section 2.4. Finally, a list of sorted documents is created using the extent of the cousin concepts. A document is only inserted in the list once, from the cousin concept with the highest rank. The order of the documents inserted in the list from the same cousin concept is disregarded.

Name	#documents	#terms	#annotations	Ctx. density	# queries
CISI	1460	8169	68827	0.05%	30
CACM	3204	7466	67502	0.2%	53
MED	1033	11207	57370	0.4%	26
CRAN	1398	5964	77743	0.9%	100

Table 2.1: Dataset characteristics

In order to compare the results of our approach, we have implemented three retrieval methods, namely exact matching, BM25 and CLR. The *exact matching* (EM) method is a naive approach which searches the database for documents with at least one keyword provided in the query. Additionally, documents are ranked according to how many terms they have in common w.r.t. the query. Documents with more terms in common are ranked first. The BM25 function [93] (also known as Okapi BM25) uses a probabilistic approach to rank documents considering collection size and document length normalization. Each document is *scored* w.r.t. the query using a modified and parametric version of term-frequency and inverse-document-frequency (TF.IDF) weighting scheme. BM25 is within the BM (Best Match) family of retrieval methods that are less restrictive than EM methods. CLR (concept lattice-based ranking) corresponds to the standard lattice-based approach using neighbourhood expansion [24], as explained in Section 2.3.2.

2.5.2 Evaluation measures

In the following, we provide a description of the evaluation measures used in this work as described in [93]. Precision and Recall are measures that assess the relevance of documents answered

[33] `http://nltk.org`
[34] `http://networkx.github.com`

by a retrieval system. Formally, given a query `q`, we define precision and recall in Equations 2.3 and 2.4 (respectively) where the set *retrieved* contains all documents found for `q`, and the set *relevant* contains the documents which constitute the actual answer for `q` (ground truth). The set $positive = |retrieved \cap relevant|$ represents the correct subset of the retrieved documents for query `q`.

$$precision(retrieved) = \frac{|positive|}{|retrieved|} \tag{2.3}$$

$$recall(retrieved) = \frac{|positive|}{|relevant|} \tag{2.4}$$

The *relevant* set (or "ground truth") is usually constructed by a single or a group of domain experts which are able to distinguish within the document collection which are the relevant documents for a given query (sometimes checking documents one by one). Having the ground truth, the calculation of precision and recall is straightforward. For example, consider the query with keywords "arthroscopy" and "complication" answered by CLAIRE with documents shown in list 1 at Table 2.2. These documents correspond to the *retrieved* set (example illustrated in Figure 2.7). The *relevant* set is defined in list 2 at the same Table. From this we obtain that CLAIRE is able to retrieve 3 out of 4 documents that domain experts considered relevant and thus, our query was answered with a recall of 0.75. However, along with those 3 relevant documents, our system also found 2 documents not considered by domain experts which are regarded as false-positives. Hence, our system finds 3 correct documents out of 5 which yields a precision of 0.6.

List	Set Name	Documents	Precision	Recall
1	Ground Truth	d_4, d_6, d_7, d_9	-	-
2	System Answer	d_2, d_6, d_7, d_8, d_9	0.6	0.75
3	Low Quality/High Quantity	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$	0.44	1
4	High Quality/Low Quantity	d_7	1	0.25
5	System answer (ranked)	d_7, d_8, d_9, d_2, d_6	0.6	0.75

Table 2.2: Precision and Recall - Examples

Row	Document	Relevant?	Precision	Recall
1	d_7	✓	1	0.25
2	d_8	✗	0.5	0.25
3	d_9	✓	0.66	0.5
4	d_2	✗	0.5	0.5
5	d_6	✓	0.6	0.75

Table 2.3: Values of precision and recall calculated considering the first 1,2,3,4 and 5 ranked elements in list 5 at Table 2.2. Precision and recall are calculated with a list which considers all previous documents in decreasing order of ranking (for example, precision and recall in the third row were calculated for documents d_7, d_8 and d_9. Column "Relevant?" shows if the document in the corresponding ranking is relevant (in list 1 at Table 2.2).)

In a nutshell, *precision* measures the proportion of correct answers among the answers found by a retrieval system. Hence, precision takes into consideration also the incorrect answers of the system. On the other hand, *recall* measures the proportion of correct documents over the whole

collection of possible correct answers for a given query and does not consider incorrect answers. Precision and recall are often considered as a trade-off between the quality and the quantity of the answers where a high quality is achieved with a high precision value, and a high quantity of correct answers is achieved with a high recall value. Actually, a high recall value for a given user query can be easily achieved by retrieving the whole set of documents in the collection. However, this comes at the cost of a low precision (low quality/high quantity). For example, consider in Table 2.2 list number 3 with recall of 1 and precision of 0.44. Inversely, it is easy to achieve a high precision by retrieving a few documents which are likely to be correct at the cost of a low recall (high quality/low quantity). In Table 2.2, this is the case of list 4 with precision of 1, but recall of 0.25.

As discussed in Section 2.4.1, it is well accepted that a good retrieval system should maintain a balance between quality and quantity (precision and recall), however if we want to compare CLAIRE w.r.t. other systems, we need something more robust to draw conclusions. Another aspect of precision and recall is that they do not take into consideration the document ranking in the answer of a retrieval system. For example, consider in Table 2.2 lists 2 and 5 (not ranked and ranked, respectively) which have the same values of precision and recall. We do not only need an evaluation for the answers, but also for the ranking applied to the answers.

Regarding these drawbacks, some other evaluation measures have been proposed considering precision and recall in the context of ranking. Table 2.3 shows the ranked answer in list 5 at Table 2.2 where precision and recall were calculated considering only the first 1, 2, 3, 4 and 5 documents of the ranking. We can appreciate that different values of precision are obtained depending on how many documents are considered in the answer (usually, because of the quality/quantity trade-off, the tendency is that as more documents are considered, less precision is achieved).

In the following, we define the evaluation measures for a ranking process considered in this work. These measures are used as standard IR evaluation techniques [6].

Considering the precision and recall values in Table 2.3, we define the *interpolated precision at a recall interval* as the maximum value of precision achieved with a recall value in that interval. For example, the interpolated precision at the recall interval $[0.5, 1]$ is the maximum among $0.66, 0.5, 0.6$ (for rows 3,4 and 5 in Table 2.3, respectively), and hence is 0.66. Similarly, the interpolated precision in the recall interval $[0, 0.5]$ is 1. The *interpolated average precision* (IAP) is the mean of the interpolated precisions for all recall intervals. In this example, is the mean of the interpolated precisions for $[0, 0.5]$ and $[0.5, 1]$ and hence, it is 0.83. For two intervals, we say 2-point interpolated average precision or IAP@2. The usual approach considers 11 intervals and it is called 11-point interpolated average precision or IAP@11.

Analogous to IAP, we calculate the *mean average precision* (MAP) which is the mean of the precision values in Table 2.3 where the column "Relevant?" is marked as correct (✓). This is, it only considers the precision values where the last document of the list (d_x) is a relevant document[35]. The MAP in this example is the average of the precision values of rows 1,3 and 5 which is 0.753.

To formalize these measures, let us define the set $rank = \{d_1, d_2, ..., d_n\}$ as the list of documents answered by a retrieval system for a given query sorted by descendent ranking. We define $rank^{d_x} \subseteq rank$ as a sub-list of $rank$ which contains all documents from the first one until element $d_x \in rank$. Equation 2.5 is the interpolated precision in a given interval defined by the edges r_1 and r_2. Equation 2.6 defines the 11-point interpolated average precision and Equation

[35]To be strictly correct, this is called the "average precision" (AP), while MAP is the mean of AP over a set of queries. Since we are presenting all of our results averaged over a set of queries, in this work we refer to AP as MAP.

2.7 describes the *mean average precision* as used in this work.

$$ip(r_1, r_2) = \underset{d_x \in rank}{\text{argmax}}\{precision(rank^{d_x}) \iff recall(rank^{d_x}) \in [r_1, r_2[\} \tag{2.5}$$

$$IAP@11 = \frac{\sum_{i=0}^{10} ip(\frac{i}{10}, \frac{(i+1)}{10})}{11} \tag{2.6}$$

$$MAP = \frac{\sum_{x \in positive} precision(rank^x)}{|positive|} \tag{2.7}$$

Finally, in this work we also provide the precision calculated in the first five documents of the ranking or P@5. This is not a measure that evaluates the ranking, but it gives an insight on the practicability of the approach, given that users tend to evaluate the retrieved set of documents by the relevance of the few first elements in the ranking. In the case of Table 2.3, the P@5 is given by the precision on list 5 and it is 0.6.

2.5.3 Results

Table 3.4 shows the results for 3 measures, namely interpolated average precision at 11-points (IAP@11), mean average precision (MAP) and precision in the first 5 ranked documents (P@5) on the four datasets and the four approaches evaluated. Values in boldface indicate the best value obtained for an approach for each dataset. From these results it can be appreciated that CLAIRE surpasses the other three approaches with a score of CLAIRE: 9, CLR: 0, EM: 2 and BM25: 1. CLAIRE always wins in the values of IAP and MAP which are actually the measures that consider document positions and same precision/recall conditions whereas it only wins once in the precision for the first five ranked documents.

This can be explained by the manner used by CLAIRE to rank documents according to the semantic similarity among the terms shared by a subset of documents and the terms in a query. A subset of documents may have a few terms in common w.r.t. the query, but several *similar* terms which will rank them high in the retrieved list compared with documents with more terms in common w.r.t. the query but a few *similar* terms. It seems that having more terms in common w.r.t. the query is more important than having more *similar* terms. On the other hand, most documents do not have many terms in common w.r.t. the query. This means that approaches like EM and BM25 are very good at ranking a few documents (those that have more terms in common with the user query) and bad at ranking many documents where their discriminatory power is low (those that have more similar terms rather than the same terms in common with the user query). The discriminatory power of CLAIRE does not decrease in this manner making it robust w.r.t. documents with different terms than those of the query, mainly by the use of semantic similarity among terms. This also explains why CLAIRE always wins in the CRAN dataset where all the low score values suggest a rather poor relation between query terms and documents.

This is further supported by Figures 2.8 which present the interpolated precision at 11 different recall points. It can be seen that, with exception of the CRAN dataset, EM always has the best value in the first point 0.0, while CLAIRE quickly surpass EM (and the other approaches) for the rest of the recall points.

Finally, regarding the use of Formica's similarity, in these experiments we did not seek for an optimal value of w since our goal was to show the feasibility of our approach and to prove that

Dataset	MED				CACM			
Approach	CLAIRE	CLR	EM	BM25	CLAIRE	CLR	EM	BM25
IAP@11	**0,5451**	0,4619	0,5116	0,5015	**0,2756**	0,1504	0,2135	0,0848
MAP	**0,5029**	0,402	0,4686	0,4804	**0,2524**	0,1697	0,1939	0,0724
P@5	0,48	0,336	0,5524	**0,5714**	0,1608	0,0769	**0,2154**	0,1038
Dataset	CISI				CRAN			
Approach	CLAIRE	CLR	EM	BM25	CLAIRE	CLR	EM	BM25
IAP@11	**0,3527**	0,2978	0,17	0,1762	**0,0279**	0,0199	0,0154	0,0181
MAP	**0,3234**	0,259	0,1444	0,1499	**0,0262**	0,0181	0,013	0,0159
P@5	0,2303	0,1886	**0,28**	0,2571	**0,0122**	0,0043	0,0043	0,0106

Table 2.4: Measures for each domain and each approach

better or similar results could be obtained with CLAIRE compared to standard IR techniques. Nevertheless, it is worth mentioning that for most queries (particularly the large ones), the query concept extent is empty, i.e. it is very hard that a document contains exactly all the terms provided in the user query. In these cases the left (additive) term of Formica's similarity is not informative (equal to 0). To avoid this, we use the heuristic of considering the query concept extent as containing the union of all its query generators' extents. This has shown to greatly improve the results in all the four datasets used in the experiments.

2.5.4 Query analysis

In the following we present a brief analysis for some queries of the CISI dataset in order to provide a deeper understanding on how CLAIRE is able to obtain better results than the other approaches. We also discuss further improvements for CLAIRE.

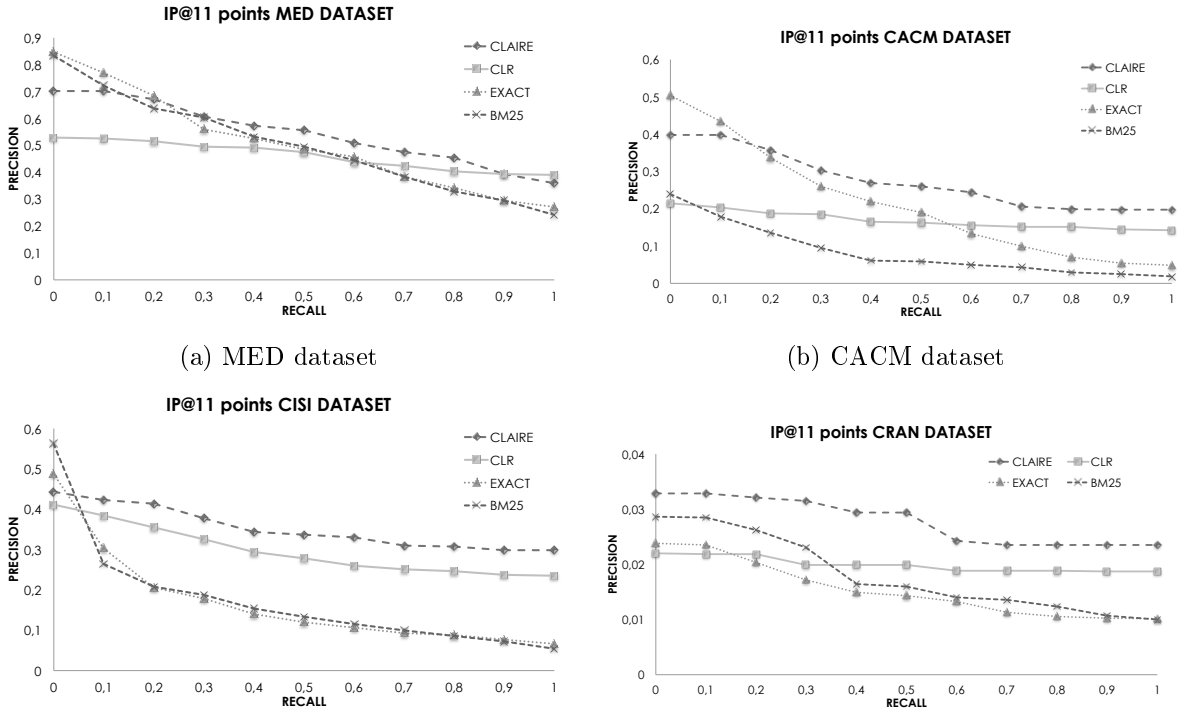
The sunny case of query 6: Query 6 contains terms communication, verbal, possibility, word, computer and human, however it is only mapped to just one relevant document (out of 1460). Our approach is able to find 30 documents in the query generators shown in Table 2.5.

ID	Intent
$G6.1$	word, computer, human
$G6.2$	computer, communication
$G6.3$	possibility, human

Table 2.5: Query generators for query 6

These three query generators led us to 35 different cousin concepts (query modifications) from which the top 10 in the ranking are shown in Table 2.6 where the query concept is also illustrated in grey. The concept with the highest similarity to the query concept (second row) contains in its extent 4 documents including the only correct answer for the query. In this case, the query modification is created by replacing the term communication (in the sense of "Something that is communicated by or to or between people or groups") by the term information (in the sense of "A message received and understood"). Thus, thanks to the search through cousin concepts, the system is able to find this unique relevant document showing the capabilities of our approach.

The infamous case of query 8: The case of query 8, consisting of terms language, indexing, information, retrieval and science, is of special interest since we are able to find 76 different query modifications with very high similarity values w.r.t. the original query (shown in Table 2.8) leading us to 31 documents which include none of the possible 18 correct answers.



(a) MED dataset

(b) CACM dataset

(c) CISI dataset

(d) CRAN dataset

Figure 2.8: 11-point interpolated precision for each dataset

The problem is due to the fact that the query generators (in Table 2.7) do not contain any correct answers making any possible query modification useless, particularly because the keywords provided in the query are too general. One possible way to overcome this issue corresponds to the inclusion of more documents by relaxing the definition of cousin concepts allowing query generators to be at a distance 2 of the query concept, however this induces an explosion on the query modifications obtained from the lattice (from 76 to 504) and a consequent lower performance of the document retrieval process.

Finding more documents: As stated above, it is possible to increase the *recall* of our approach (the number of relevant documents retrieved over the total number of relevant documents) by relaxing the definition of cousin concepts allowing the query generators to be at distances farther than 1 from the query concept. However, this comes with a great cost in terms of computation since the number of query modifications obtained from the query space (the concept lattice) which should be compared to the query concept will greatly increase. It also impacts negatively in the precision of the answers (the number of relevant documents retrieved over the total amount of documents retrieved).

Applicability: It is worth mentioning the applicability of our approach given the limitations in the computation of a concept lattice. With the state-of-the-art FCA algorithms, it is uncertain that CLAIRE may be applied in document collections such as the entire Web or even some subsets of it proposed as standard datasets for testing IR tasks[36]. Indeed, the applicability of CLAIRE is restricted to smaller datasets, usually personal data collections where the number of documents is not larger than 100.000 documents, such as personal picture collections, research references,

[36]The TReC competition provides several datasets for different IR tasks. Some of them contain billions of documents, e.g. `http://trec.nist.gov/`

Formica Sim	Intent	Extent support
1	communication, verbal, possibility, word, computer, human †	0
0.637	word, computer, human, information	4
0.634	machine, word, experiment, based, computer, human, text	3
0.603	research, word, computer, human	3
0.602	index, word, computer, human	3
0.595	word, computer, make, human	3
0.493	concept, possibility, human, analysis	3
0.484	computer, form, communication	4
0.470	possibility, human, information	3
0.470	computer, information, communication	15
0.449	computer, part, communication	4

† Grey row represents the *query concept*

Table 2.6: Ranked concepts for query 6

ID	Intent
$G8.1$	language, information, retrieval, indexing
$G8.2$	science, information, retrieval, indexing
$G8.3$	science, language, information, indexing

Table 2.7: Query generators for query 8

mail archives, music albums, etc. These datasets share three characteristics: they are real-life datasets, they are numerous since mostly any person with a computer creates several of them; and more importantly, there is a real necessity to improve the performance of searching within them.

2.6 Conclusions

Retrieval systems are complex in the sense that they involve a wide variety of techniques from different domains. For example, in this chapter we have discussed natural language processing techniques (such as lemmatization and stemming), annotation (semantic mapping), mining (FCA) and case-based reasoning. From a software engineering point of view, maintaining coherence in such a composition of components is a difficult task.

The process of knowledge discovery in databases proposes a robust framework to achieve this composition. Moreover, IR is a KDD process in its very core for two important reasons. Firstly, it is a human-centered process since it is focused on satisfying user information needs. Secondly, it requires the systematic transformation of data (from the document metadata in the corpus) to information (the retrieved ranked list of documents) to knowledge (what the user gets from documents retrieved). For these reasons, we have proposed a KDD-like approach for an IR system based on FCA called CLAIRE (Concept Lattices for Information REtrieval).

Like several systems before it, CLAIRE shows the good capabilities of FCA for IR. A concept lattice generates a "map" which connects the document space and the query space in a natural and intuitive manner. Exploiting this fact, we have shown that a more sophisticated navigation technique allows us to enhance the quality of the documents retrieved using a concept lattice. Moreover, we have shown how we can combine the lattice structure with external knowledge sources in order to leverage the retrieving potential of FCA. Indeed, a concept lattice gives us the "map" of the query space telling us which two points are connected and how. Semantic

Formica Sim	Intent	Extent cardinality
1	language, indexing, information, retrieval, science †	0
0.987	science, word, information, retrieval, indexing	2
0.977	subject, language, information, retrieval, indexing	5
0.977	language, field, information, retrieval, indexing	5
0.972	science, information, term, retrieval, indexing	4
0.959	science, subject, information, retrieval, indexing	4
0.936	method, language, information, retrieval, indexing	7
0.927	method, science, information, retrieval, indexing	4
0.926	system, language, information, retrieval, indexing	13
0.921	science, research, information, retrieval, indexing	4
0.916	concept, language, information, retrieval, indexing	3

† Grey row represents the *query concept*

Table 2.8: Ranked results for query 8

similarity provides us with a distance between these points, allowing us to decide where to go in this map.

The contributions of the work presented in this chapter are the following:

1. A KDD process as a framework for retrieval systems
2. A navigation-ranking strategy for retrieval based on concept lattices
3. The formalization of cousin concepts as a CBR-based navigation strategy
4. A semantic-similarity based evaluation for retrieval using formal concepts

The lessons learnt are:

- In Section 2.5.3 we discuss the fact that our ranking system is less precise in the first part of the ranking (P@5) due to the way semantic similarity works (e.g. 1 pair of equal terms are as important as 2 pairs of half-similar terms)
- As pointed out in Section 2.5, the cousin concepts definition can be too restrictive to find related documents for very general keywords in the query (see "The infamous case of query 8")

Both of these problems are related to the score given to terms in a document for ranking purposes. Probabilistic methods already take this phenomenon into account, for example using *inverse document frequency* which "punish" very general terms in the corpus with a lower *scoring factor*. Nevertheless, these considerations have to be taken into account in the "indexing" stage of the retrieval system which our model cannot support as it would require a complex document description, i.e. not Boolean attributes but a vector of numerical values. In the following chapter, we will discuss a new model for indexing documents using the "vector space model" of retrieval.

Chapter 3

Contributions on Indexing

Contents

3.1 Introduction 55
3.2 Background 56
3.2.1 The vector space model for retrieval (VSM) 56
3.2.2 Interval Pattern Structures 58
3.2.3 Relational Concept Analysis (RCA) 59
3.3 CLAIRE and the vector space model 61
3.3.1 Querying 62
3.3.2 Retrieving documents with ip-CLAIRE 64
3.3.3 Experimental results 64
3.3.4 Discussion on the capabilities of ip-CLAIRE 65
3.4 A model for heterogeneous indexing 66
3.4.1 Inspiring problem - Latent Semantic Indexing 67
3.4.2 Adapting RCA for pattern structures 69
3.4.3 Discussion on the heterogeneous pattern structures model 74
3.5 Conclusions 76

3.1 Introduction

As described in Section 1.4.1, FCA is a natural implementation of the Boolean IR model. Sadly, modern retrieval systems have shifted to more sophisticated models, such as the "vector space model" or the "probabilistic model". In these models, documents are not described by a set of terms but by more complex descriptors, such as vectors of numeric values or probability functions, respectively and thus, they tend to catch in a better way the relations among documents and terms. For example, in the vectorial space model we are able to measure distance between documents that in the Boolean IR model are incomparable [129]. In the TF.IDF weighting scheme (term frequency - inverse document frequency, a probabilistic model) we are able to differentiate between general and specific terms when comparing two documents [93].

This chapter is divided in two main contributions. In the first part, we introduce a FCA-based IR model supporting complex document descriptors. This is achieved by the implementation of the interval pattern structures framework [79] through which we are able to construct a concept

lattice of formal concepts with interval patterns instead of intents. Interval patterns describe convex regions in an Euclidean space, thus allowing us to implement the dynamics of the vector space model of retrieval. While experimental evaluation results show a promising application for document retrieval purposes, we realise that a new model is needed in order to support retrieval in real datasets given the computational limitations associated to computing an interval pattern concept lattice.

In the second part we move forward towards a novel indexation system in which documents are described by heterogeneous descriptors, i.e. a mixed pattern of numerical values and Boolean attributes. To achieve this, we define the heterogeneous pattern structure framework. We show how, through the implementation of this model, we can find "meaningful" regions in an Euclidean space in the form of convex regions annotated with semantic tags. Doing this, we are able to restrict the size of the concept lattice which would allow applying interval pattern structures in larger datasets.

3.2 Background

In this section we introduce some basic theoretical concepts used throughout this chapter. A more general theoretical background introduction is provided in Chapter 1.

3.2.1 The vector space model for retrieval (VSM)

Let us consider that we need to index the book "La Araucana" of Alonso de Ercilla using the following description:

> *"La Araucana (also known in English as The Araucaniad) is a 16th-century epic poem in Spanish about the Spanish Conquest of Chile by Alonso de Ercilla. It was considered the national epic of the Kingdom of Chile and one of the most important works of the Spanish Golden Age (Siglo de Oro)."*
>
> English wikipage on La Araucana[37]

A naive approach would be to analyse this text into a set of terms which can be later used in the Boolean indexing process we have previously discussed. Thus, we have the bag-of-words with terms:

araucana, english, araucaniad, 16th-century, epic, poem, spanish, spanish, conquest, chile, alonso, ercilla, national, epic, kingdom, chile, important, work, spanish, golden, age, siglo, oro

Notice that in this list the terms english and spanish are both equally important indexing elements for a Boolean retrieval system and thus, given the query english epic poem, this book would be retrieved even when it was written in spanish.

A main problem with the Boolean retrieval model is that we are not able to distinguish the role that terms play in a document description. For example, it is clear that the term spanish is more important than english for this specific book, a fact that, aside from the grammatical point of view, can be inferred by the number of times the term is mentioned in the text (3 times for spanish, versus 1 time for english). This is also true for terms epic and chile with two mentions each.

[37] http://en.wikipedia.org/wiki/La_Araucana

In the VSM, we can represent the "importance" of a term w.r.t. a given document by a numerical value which we call "weight". Thus, it is necessary to define a "weighting function" $w : (D \times M) \to \mathbb{R}$ which assigns for a given document-term pair (d, m) a weight value $\mathtt{w}$ based on an *arbitrary* notion of the "importance" of m within d. We stress the *arbitrary* character of the weighting function since its definition strongly depends on the dataset characteristics and the application domain of the retrieval system. In [93], several weighting functions (or schemes) are discussed using different rationales. Given that our model is independent of the weighting function used, we will perform a simple "term frequency" weighting.

Let m be a term within document d, its frequency $tf_{d,m}$ is given by the number of occurrences of m in d (denoted by $n_{d,m}$, e.g. $n_{\text{ARAUCANA,spanish}} = 3$. Notice tha tin this case, ARAUCANA is a document, not a term) over the length of d. The length of a document comprises all different terms and their different occurrences, e.g. the length of document ARAUCANA is 23. More formally, term frequency is defined in Equation 3.1.

$$tf_{d,m} = \frac{n_{d,m}}{\sum\limits_{m_i \in M} n_{d,m_i}} \tag{3.1}$$

Thus, we can obtain that $tf_{\text{ARAUCANA,spanish}} = 3/23 \equiv 0.13$.

In the VSM model, a document is described by a vector of term-weights in a canonical order, i.e. each dimension corresponds to a single term which is the same for all documents. For instance, let us re-use the example formal context $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ in Table 1.8 of Section 1.4.1. We will consider each relation $\mathtt{gIm}$ as a single occurrence of term $\mathtt{m}$ in document $\mathtt{g}$ or what is the same, $(\mathtt{g}, \mathtt{m}) \in \mathtt{I} \implies n_{d,m} = 1$. Thus, we can construct the many-valued formal context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$ showed in Table 3.1 (notice that terms in the context are assigned with a numbered attribute label $\mathtt{m}_i, i \in [1, 12]$), where

$$tf_{\mathtt{g},\mathtt{m}} = \mathtt{m}(\mathtt{g}) = \frac{1}{|\mathtt{g}'|}$$

For example, $\mathtt{g}'_6 = \{\mathtt{m}_9, \mathtt{m}_{11}\}$, then $|\mathtt{g}'_6| = 2$ and $tf_{\mathtt{g}_6,\mathtt{m}_{11}} = 0.5$. Given a weighting function such as the one defined in Equation 3.1, the vectorial representation of a document $\mathtt{g}$ is defined as:

$$\vec{\mathtt{g}} = \langle tf_{\mathtt{g},\mathtt{m}_i} \rangle_{i \in [1,|\mathtt{M}|]}$$

Thus, a vectorial representation for a document in the many-valued context of Table 3.1 is given by its respective row, for example:

$$\vec{\mathtt{g}_1} = \langle 0.25, 0.25, 0.25, 0, 0, 0, 0, 0, 0, 0.25, 0, 0 \rangle$$

We can observe how the set of terms $\mathtt{M}$ defines a vectorial space with a number of dimensions equal to its cardinality where each document d is represented as a point with coordinates $d(i)$ defined by the weighting function evaluated for d and the term $\mathtt{m}_i$. Using this paradigm, we can resort to a distance measure in order to evaluate document similarity, i.e. the closer two documents are in the space, the more similar they are. Examples of such measures are the Euclidean distance and the Cosine distance [45].

	patient	laparoscopy	scan	user	medicine	response	time	MRI	practice	complication	arthroscopy	infection
	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$	$\mathtt{m}_5$	$\mathtt{m}_6$	$\mathtt{m}_7$	$\mathtt{m}_8$	$\mathtt{m}_9$	$\mathtt{m}_{10}$	$\mathtt{m}_{11}$	$\mathtt{m}_{12}$
$\mathtt{g}_1$	0.25	0.25	0.25	0	0	0	0	0	0	0.25	0	0
$\mathtt{g}_2$	0	0	0.16	0.16	0.16	0.16	0.16	0	0.16	0	0	0
$\mathtt{g}_3$	0	0.25	0	0.25	0.25	0	0	0.25	0	0	0	0
$\mathtt{g}_4$	0.3	0	0	0	0.3	0	0	0.3	0	0	0	0
$\mathtt{g}_5$	0	0	0	0.3	0	0.3	0.3	0	0	0	0	0
$\mathtt{g}_6$	0	0	0	0	0	0	0	0	0.5	0	0.5	0
$\mathtt{g}_7$	0	0	0	0	0	0	0	0	0	0.5	0.5	0
$\mathtt{g}_8$	0	0	0	0	0	0	0	0	0	0.3	0.3	0.3
$\mathtt{g}_9$	0	0	0	0	0	0	0	0	0	0	0.5	0.5

Table 3.1: Many-valued formal context of term frequencies in each document.

Finally, it is worth noticing that a document corpus dataset as the one illustrated in Table 3.1 can be represented in different ways, namely as a many-valued formal context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, as a set of triples $(\mathtt{g}, \mathtt{m}, tf_{\mathtt{g},\mathtt{m}})$ or as a *document-term matrix* $A_{ij} = tf_{\mathtt{g}_i,\mathtt{m}_j}$. The reason behind these different representations is that they are useful in different context of applications. In the following sections when using the many-valued formal context or the document-term matrix representation, they will be differentiated by their denomination.

3.2.2 Interval Pattern Structures

The "interval pattern structures" setting is a pattern structure instance (see Section 1.2.6) introduced in [79] for numerical analysis purposes using the FCA framework. In this setting an object description $\mathtt{g} \in \mathtt{G}$ is a vector of intervals defined as $\delta(\mathtt{g}) = \langle[\mathtt{l}_\mathtt{i}, \mathtt{r}_\mathtt{i}]\rangle$ with $\mathtt{i} \in [1..|\mathtt{M}|]$, $\mathtt{l}_\mathtt{i}, \mathtt{r}_\mathtt{i} \in \mathbb{R}$ and $\mathtt{l}_\mathtt{i} \leq \mathtt{r}_\mathtt{i}$.

Recall from Section 1.2.6 that, in order to define a pattern structure setting, the space $\mathtt{D}$ has to be ordered w.r.t. a similarity operator $\sqcap$ which in turn allows defining the semi-lattice of object description $\underline{\mathtt{D}}$. In this case, for two interval patterns $\mathtt{d}_1 = \langle[\mathtt{l}^1_\mathtt{i}, \mathtt{r}^1_\mathtt{i}]\rangle$ and $\mathtt{d}_2 = \langle[\mathtt{l}^2_\mathtt{i}, \mathtt{r}^2_\mathtt{i}]\rangle$, the similarity operator is defined in Equation 3.2.

$$\mathtt{d}_1 \sqcap \mathtt{d}_2 = \langle[min(\mathtt{l}^1_\mathtt{i}, \mathtt{l}^2_\mathtt{i}), max(\mathtt{r}^1_\mathtt{i}, \mathtt{r}^2_\mathtt{i})]\rangle \tag{3.2}$$

For a more intuitive notion on $\underline{\mathtt{D}}$, consider interval pattern vectors with a single dimension (i.e. with a single interval) $\langle[1,1]\rangle, \langle[2,2]\rangle, \langle[3,3]\rangle, \langle[4,4]\rangle$ and their semi-lattice representation in Figure 3.1.

Finally, an interval pattern concept $(\mathtt{A}, \mathtt{d})$ defines a convex region within the given description space (represented by $\mathtt{d}$) and a set of objects *populating* that region (represented by $\mathtt{A}$). This fact will be of great relevance in the following sections since it allows us to generate "clusters" of documents[38].

[38] Hereafter we use the notion of "cluster" as a convex region in the LV-space grouping a set of documents.

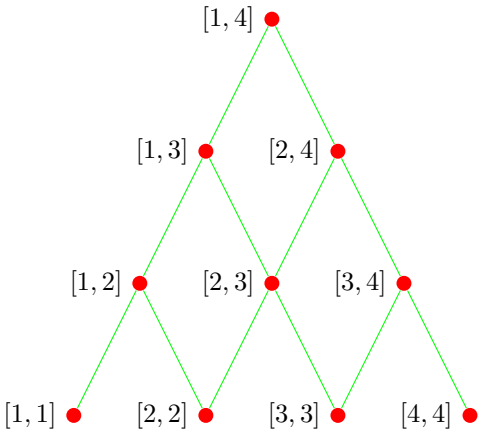


Figure 3.1: A semi-lattice representation of intervals

3.2.3 Relational Concept Analysis (RCA)

As we have previously discussed, FCA-based IR approaches have been widely used for enriching document descriptions through external knowledge sources in order to improve the quality of the answers given for querying users (see Section 1.4.3). The enrichment process has been performed in different ways, either by creating a set of many-valued contexts [120] or by modifying the standard FCA closure operator [21]. Following these lines, relational concept analysis (RCA) [126, 127] was proposed as an extension of FCA defining a process for the iterative enrichment of object descriptions exploiting relations between objects (not just between objects and their attributes) based on the notion of "relational scaling".

Consider our running example, the document-term formal context shown in Table 1.8 (repeated in Table 3.2c). Until now, we have described this table as the relation between a set of objects (documents) and their attributes (terms). However, we can consider that terms are also objects with attributes of their own. For example, Table 3.2b shows the relation between these terms and a set of six different *synonyms* (or synsets) extracted from Wordnet. In this table, we can consider that terms have the role of objects while synsets are attributes of the terms.

Table 3.2a shows a formal context where documents are now related to a different set of attributes, this time the authors that created them[39]. In this new setting, we will consider that the initial formal context of documents and terms is now a *relational context* which describes the connection between two sets of objects, namely the set of documents $\mathtt{G}_1$ and the set of terms $\mathtt{G}_2$.

While the new model may seem as a simple change of semantics in the variables involved, it is actually much deeper than that. In contrast with FCA, our goal using RCA is to be able to index documents w.r.t. formal concepts of terms and synsets instead of just plain terms. This is, we would like to find groups of documents containing an *abstract description* which can be characterized by synsets. For example, we can consider the following abstract description:

A set of documents containing terms which refer to the sense of Person

In this description the terms are not important, as long as they refer to the sense Person. This is the case for documents $\mathtt{g}_1$,$\mathtt{g}_2$,$\mathtt{g}_3$,$\mathtt{g}_4$ and $\mathtt{g}_5$ which contain terms patient or user, while in the initial formal context of Table 3.2c this extent (containing all five documents) does not exist.

[39] Actually, authors can also be considered as objects as they clearly have attributes of their own, e.g. nationality, language, period, etc. For the sake of simplicity, in this example we consider them as simple document attributes.

	author1	author2	author3	author4
g_1		×	×	
g_2	×	×		
g_3	×	×		
g_4	×	×		×
g_5			×	
g_6				×
g_7			×	×
g_8			×	
g_9	×			×

(a)

	Person	Surgery	Illness	Artefact	Event	Activity
patient	×					
laparoscopy		×				×
scan				×		
user	×					
medicine						×
response					×	
time					×	
MRI				×		
practice						×
complication			×			
arthroscopy		×				×
infection			×			

(b)

	patient	laparoscopy	scan	user	medicine	response	time	MRI	practice	complication	arthroscopy	infection
g_1	×	×	×							×		
g_2			×	×	×	×	×		×			
g_3		×		×	×			×				
g_4	×				×			×				
g_5				×		×	×					
g_6									×		×	
g_7										×	×	
g_8										×	×	×
g_9											×	×

(c)

Table 3.2: Relational context family (RCF) - Table 3.2a: Formal context $\mathcal{K}_1$ of documents and their authors. Table 3.2b: Formal context $\mathcal{K}_2$ of terms and their Wordnet annotations. Table 3.2c: Relational context $\mathtt{aw}$ representing the relation *document "annotated with" term*

In the following, we provide a formalization for this model using the notation in [126, 127] and the formal contexts shown in Table 3.2, where Table 3.2c is called a *relational context* (instead of a formal context, since it describes the relation between two sets of objects) and represents the relation "document *annotated with* term" (denoted as $\mathtt{aw}$).

A *relational context family* (RCF) is a set of contexts $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2\}$ and a set of binary relations $\mathbf{R} = \{\mathtt{r}\}$[40]. A relation $\mathtt{r} \subseteq \mathtt{G}_1 \times \mathtt{G}_2$ connects two object sets, a *domain* $\mathtt{G}_1$, ($\mathtt{dom}(\mathtt{r}) = \mathtt{G}_1$) and a *range* $\mathtt{G}_2$, ($\mathtt{ran}(\mathtt{r}) = \mathtt{G}_2$). Moreover, a relation $\mathtt{r}$ can be seen as a set-valued function $\mathtt{r} : \mathtt{G}_1 \rightarrow \wp(\mathtt{G}_2)$ [126].

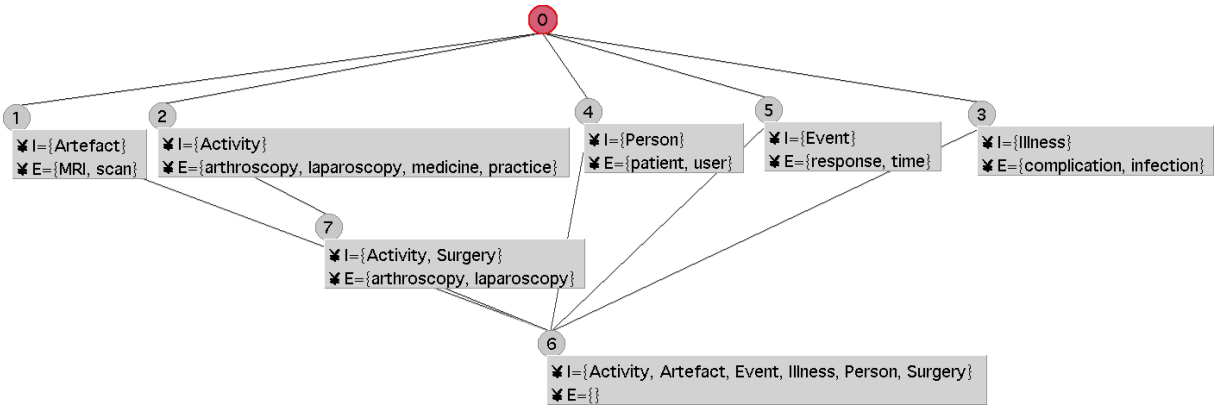


Figure 3.2: Concept lattice of formal context $\mathcal{K}_2$ in Table 3.2b

For the current example, let $\mathtt{G}_1$ be a set of documents and $\mathtt{G}_2$ be a set of terms. Then the corresponding RCF is composed by contexts $\mathcal{K}_1 = (\mathtt{G}_1, \mathtt{M}_1, \mathtt{I}_1)$ (with $\mathtt{M}_1, \mathtt{I}_1$ as shown in Table 3.2a), $\mathcal{K}_2 = (\mathtt{G}_2, \mathtt{M}_2, \mathtt{I}_2)$ (with $\mathtt{M}_2, \mathtt{I}_2$ as shown in Table 3.2b) and the relational context $\mathtt{aw}$ in Table 3.2c.

RCA is based on a *relational scaling* mechanism that transforms a relation $\mathtt{r}$ into a set of

[40] Actually, a relational context family is defined to contain $n \geq 2$ contexts in $\mathbf{K}$ and $n-1$ relations in $\mathbf{R}$. However, since all relations are binary, it is sufficient and simpler8 to define our approach in the described setting.

	author1	author2	author3	author4	∃aw : C1	∃aw : C2	∃aw : C3	∃aw : C4	∃aw : C5	∃aw : C6	∃aw : C7
g_1		×	×		×	×	×	×			×
g_2	×	×			×	×		×	×		
g_3	×	×			×	×		×			×
g_4	×	×		×	×	×		×			
g_5			×					×	×		
g_6				×		×	×				×
g_7			×	×		×	×				×
g_8			×			×	×				×
g_9	×			×		×	×				×

Table 3.3: Context $\mathcal{K}_1$ after relational scaling using existential quantifier. We have removed the relational attribute $\exists \mathtt{aw} : \mathtt{C0}$ usually assigned to every object

relational attributes that are added to complete the "initial context" describing the object set $\mathtt{G}_1 = \mathtt{dom}(\mathtt{r})$. For each relation $\mathtt{r}$, there is an *initial lattice* for each object set, i.e. $\mathcal{L}_1$ for $\mathtt{G}_1$ and $\mathcal{L}_2$ for $\mathtt{G}_2$.

The RCA mechanism starts from two initial lattices, $\mathcal{L}_1$ and $\mathcal{L}_2$, and builds a series of intermediate lattices by gradually completing the initial context $\mathcal{K}_1$ with new "relational attributes". Relational scaling follows the description logics (DL) semantics of role restrictions [5]. A relational attribute $\exists \mathtt{r} : \mathtt{C}$, $\mathtt{C}$ being a concept and $\exists$ the existential quantifier, is associated to an object $\mathtt{g} \in \mathtt{G}$ whenever $\mathtt{r}(\mathtt{g}) \cap \mathtt{extent}(\mathtt{C}) \neq \emptyset$ (other quantifiers are available, see [126]). The series of intermediate lattices converges towards a "fixpoint" or "final lattice" and the RCA mechanism is terminated. This is why there is one initial and one final lattice for each context of the considered RCF. For the running example, the lattice (in this case initial and final) in Figure 3.2 for the formal context $\mathcal{K}_2$ in Table 3.2b, along with the "relational context" in Table 3.2c, indicates the "relational attributes" that should be added to the formal context in Table 3.2a. For instance, using the existential quantifier, the relational attribute $\exists \mathtt{aw} : \mathtt{C4}$ ($\mathtt{C4}$ is the concept with intent Person in Figure 3.2) should be added to all documents $\mathtt{g}_i \in \mathtt{G}_1$ in formal context $\mathcal{K}_1$ in Table 3.2a if $\mathtt{g}_i$ contains terms patient or user in the relational context of Table 3.2c. Table 3.3 shows formal context $\mathcal{K}_1$ after the relational scaling process.

3.3 CLAIRE and the vector space model

In this section we will introduce a new model of CLAIRE modified to retrieve documents using the dynamics of the vector space model described above. Given a document corpus represented as a many-valued formal context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, we will define the interval pattern structure $\mathcal{K} = (\mathtt{G}, \underline{\mathtt{D}}, \delta)$ where the mapping δ is defined in Equation 3.3.

$$\delta(\mathtt{g}) = \langle [tf_{\mathtt{g},\mathtt{m}_i}, tf_{\mathtt{g},\mathtt{m}_i}] \rangle,\ \mathtt{i} \in [1, |\mathtt{M}|] \tag{3.3}$$

For example,

$$\delta(\mathtt{g}_1) = \langle [0,25,0,25],[0,25,0,25],[0,25,0,25],[0,0],\\ [0,0],[0,0],[0,0],[0,0],[0,0],[0,25,0,25],[0,0],[0,0]\rangle$$

Thus, an interval pattern concept $(\mathtt{A},\mathtt{d})$ corresponds to a convex region in the vector space of descriptions determined by $\mathtt{d}$, and a set of documents $\mathtt{A}$ with a representation in that region. Given that in the VSM, documents that are "closer" are considered similar, an interval pattern concept represents a group of documents clustered by similarity. This similarity can be *upper bounded* by measuring the Euclidean distance between the boundaries of the convex region.

It is not easy to understand the above definitions in a vector space of more than three dimensions given that we are restricted to our own three-dimensional way of thinking. Let us then use a toy example to visualize this model in a one-dimensional setting. Consider a single dimension for document descriptions, namely the dimension *age* indicating "how old" a document is in a scale of $1-4$. A value of 1 indicates that a document is very old while 4 indicates that it is very recent. Now, we can think of the *most granular* classification of documents, this is a single cluster for each value. It is easy to see that within these clusters documents are at distance of 0 w.r.t. their *age*, i.e. they all have the same value. The next *less-granular* classification contains intervals $[1,2]$, $[2,3]$ and $[3,4]$. We can label these intervals as "old documents", "modern documents" and "new documents", respectively. In this classification, documents are at most at distance 1, although some of them are at distance 0. Next, we have $[1,3]$ and $[2,4]$ which we can call "old documents" and "new documents" with a distance of at most 2. Finally, we have a single cluster $[1,4]$ containing all documents in the collection which of course, are at most at distance 3. This classification can be modelled by the semi-lattice depicted in Figure 3.1.

A consideration to be taken into account in this model is that clusters overlap, meaning that a document can be "old" and "new" at the same time (using intervals $[1,3]$ and $[2,4]$). While this may present problems for interpreting the classification, it is necessary to consider the applicability of this kind of models. For example, "La Araucana" written in the 16th century is old compared to "Harry Potter" (written in the 21st century) but quite new compared to the Iliad, dated around the 8th century B.C.

In two or more dimensions, documents are put together in convex regions defined by an interval vector $\langle[\mathtt{l}_\mathtt{i},\mathtt{r}_\mathtt{i}]\rangle, \mathtt{i} \in [1,|\mathtt{M}|]$ which are at most at distance:

$$\sqrt{\sum_{i\in[1,|M|]}(\mathtt{r}_\mathtt{i}-\mathtt{l}_\mathtt{i})^2}$$

3.3.1 Querying

Like in standard CLAIRE, we will consider a query $\mathtt{q} = \{t_1, t_2, ..t_{|q|}\}$ as a "virtual object" which can be classified by the interval pattern concept lattice derived from the document-term matrix A. Consequently, we will apply the mapping function δ to obtain its interval pattern representation using the same weighting function used to represent documents. For example, consider the query $\mathtt{q} = \{\mathsf{complication}, \mathsf{arthroscopy}\}$ for which we have:

$$\delta(\mathtt{q}) = \langle [0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0.5,0.5],[0.5,0.5],[0,0]\rangle$$

The *query concept* is still considered as the *object concept* of $\mathtt{q}$. However, the semantics of the *query space* changes. While in the Boolean retrieval model the *query space* represents a

pool of Boolean query possibilities, here the *query space* is a vectorial space divided in a set of overlapping ordered convex regions in which the query concept $(\mathtt{q}, \delta(\mathtt{q}))$ is the smallest region containing the query and all documents at 0 distance from the query.

Recall from the "hierarchical expansion" (HE) strategy explained in Section 2.3.2 the mechanism in which documents were ranked using the super-concepts of the query concept in a progressive manner. In the current setting, HE has the same meaning but different semantics. As we have pointed out, documents in the query concept extent are at distance 0 to the query representation (w.r.t. the Euclidean distance in the vectorial space). Direct super-concepts of the query concept contain documents further away from the query representation, and its own super-concepts contain documents even further away.

For instance, in Figure 3.3 we have the query $\mathtt{q}$ = {complication, arthroscopy} classified in the interval pattern concept lattice derived from the many-valued formal context in Table 3.1. For the sake of simplicity, interval patterns only contain two dimensions, one for each term in the query respectively. In Figure 3.3, the bottom concept is the query concept (marked as $\mathtt{q}$), the extent of which contains document $\mathtt{g}_7$, since $\delta(\mathtt{q})^{\square} = \{\mathtt{q}, \mathtt{g}_7\}$. It is worth noticing that the Euclidean distance between $\vec{\mathtt{g}_7}$ and $\vec{\mathtt{q}}$ is 0. The super-concepts of concept $\mathtt{q}$ contain documents $\mathtt{g}_6$ and $\mathtt{g}_9$ (concept 1) and document $\mathtt{g}_8$ (concept 2). These are the next "closer" documents w.r.t. the Euclidean distance in the vectorial space, $\mathtt{g}_6$ and $\mathtt{g}_9$ at distance 0.5, and $\mathtt{g}_8$ at distance 0.28. To enlarge the answer, we can obtain document $\mathtt{g}_1$ from concept 3, a super-concept of concept 2, which is at distance 0.56 from the query. Finally, the top concept contains all documents in the collection.

From the above example, it is clear that we can exploit the interval pattern concept lattice to retrieve and rank documents using the dynamics of the VSM, however it remains unclear how to differentiate between two formal concepts that are "in the same level" in super-hierarchy of the query concept, e.g. how can we decide if either concept 1 or concept 2 is more relevant to the query given the interval pattern concept lattice in Figure 3.1.

As we have described, document $\mathtt{g}_8$ is closer to the query than documents $\mathtt{g}_6$ and $\mathtt{g}_9$, but this is an information we have by measuring their distances using their vectorial representations $\vec{\mathtt{g}_8}, \vec{\mathtt{g}_6}$ and $\vec{\mathtt{g}_9}$. In short, this information *is lost* in the interval pattern concept lattice and thus, we cannot calculate single distances. This is not an accident. In fact, the usefulness of the concept lattice in this aspect is that we can derive the ranking information avoiding the computation of the distance between each pair of documents. Instead, as already discussed, we have *upper*

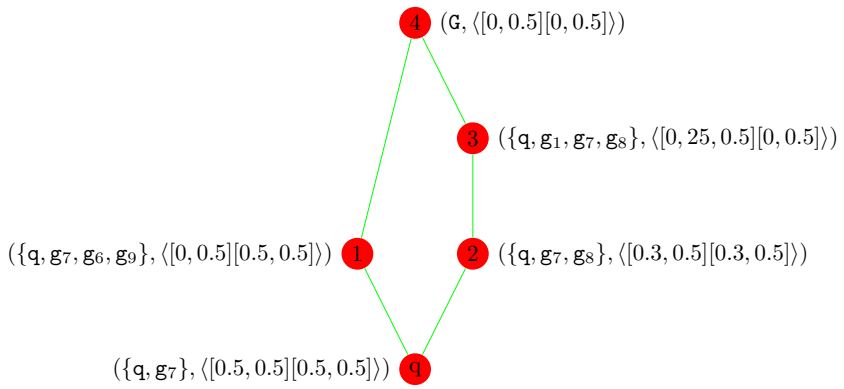


Figure 3.3: Interval pattern concept lattice derived from Table 3.1 classifying query $\mathtt{q}$ = {complication, arthroscopy}

bounds which we can compute using the boundaries of the interval pattern. The boundaries of an interval pattern $\langle[\mathtt{l_i}, \mathtt{r_i}]\rangle, \mathtt{i} \in [1, |\mathtt{M}|]$ are a couple of vectors containing its left values $\mathtt{l_i}$ and right values $\mathtt{r_i}$, and the upper bound is given by the Euclidean distance between both vectors. Consider concept 2 in Figure 3.3 with interval pattern $\langle[0.3, 0.5], [0.3, 0.5]\rangle$. Its boundaries are vectors $\langle 0.3, 0.3\rangle$ and $\langle 0.5, 0.5\rangle$ and thus, the upper bound is:

$$\sqrt{(0.3 - 0.5)^2 + (0.3 - 0.5)^2} \approx 0.28$$

This indicates that any document within this region is at most at distance 0.28 and, given that the virtual object $\mathtt{q}$ is within that region, in particular, all these documents are at most at distance 0.28 from the query. Since the upper bound of concept 1 in Figure 3.3 is 0.5, we rank document $\mathtt{g}_8$ over $\mathtt{g}_6$ and $\mathtt{g}_9$. Documents in the same concept remain undistinguishable for ranking purposes.

3.3.2 Retrieving documents with ip-CLAIRE

Finally, we can define the retrieval process using CLAIRE and the VSM paradigm which we call ip-CLAIRE, which stands for Interval Pattern Concept Lattices for Information REtrieval. The process is defined as follows:

1. Given a query $\mathtt{q}$, obtain its interval pattern representation using the mapping function δ
2. Classify $\mathtt{q}$ within the interval pattern concept lattice by computing the query concept $(\delta(\mathtt{q})^{\square}, \delta(\mathtt{q}))$
3. Obtain the super-concepts of the query concept
4. Rank formal concepts using the upper bound notion for the Euclidean distance

3.3.3 Experimental results

To test the validity of ip-CLAIRE, we applied it on a popular IR dataset which is openly available. The CISI dataset[41] consists of 1460 documents and 35 queries, each one containing a set of valid answers. Documents contain text in natural language and queries are given as a set of terms connected by Boolean operators. In our experiments, we converted documents to vectors of weighted terms stored in a relational database. The weighting scheme used was term frequency-inverse document frequency (*tf.idf*) [93]. Boolean operators in the query were disregarded since they do not provide meaning in the vector-space model (except in the extended Boolean model case [128] which was not considered for this work). The virtual query-document was constructed using the inverse document frequencies calculated from the dataset for each of its terms.

After receiving a query, ip-CLAIRE consults the database and extracts all documents that contain at least 2 terms of the query. The *query concept* is computed and classified in the lattice and its superconcepts are retrieved and ranked using the Euclidean distance between the boundaries of their interval patterns. Cosine distance (instead of Euclidean distance) was also calculated showing better results. Experiments were performed on an Intel Xeon machine running at 2.27 GHz with 62 GB of RAM memory. Table 3.4 shows the results for 11-point precision of fixed recall and 6 measures of precision for the top 5,10 and 20 ranked documents retrieved. Results on an implementation based on concept lattice-based ranking (CLR) [24] is reported along with our results for comparison purposes.

[41] `http://ftp.cs.cornell.edu/pub/smart/cisi/`

	ip-CLAIRE	CLR	EM
11-point IAP[a]	**0.232**	0.191	0.174
MAP[b]	**0.202**	0.163	0.145
Precision@5	0.257	0.206	**0.285**
Precision@10	0.251	0.174	**0.257**
Precision@20	**0.245**	0.174	0.207
Recall@5	0.032	0.049	**0.057**
Recall@10	0.060	0.073	**0.079**
Recall@20	**0.146**	0.112	0.123

Table 3.4: CISI dataset - Results for 35 queries

[a]Interpolated average precision
[b]Mean average precision

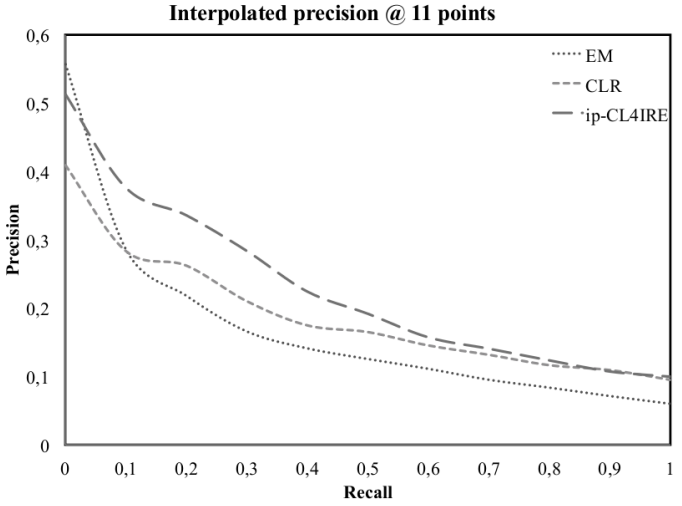


Figure 3.4: Interpolated precision in 11 points of recall

Table 3.4 reports the results in 8 measures for ip-CLAIRE, a reported CLAIRE system called concept lattice-based ranking (CLR) [24] and a naive approach called exact matching (EM) where documents are ranked according to how many terms they have in common w.r.t. the query. The second row contains the values of the interpolated average precision (IAP) over 11-points of recall illustrated in Figure 3.4. Interpolated precision in a given recall point r_i in Figure 3.4 indicates the best precision value in the interval $[r_i, r_{i+1}[$. From Figure 3.4, the interpolated precision in the recall point 0 for ip-CLAIRE is the best precision obtained in the recall interval $[0, 1[$ equal to 0.51. The third row contains the values of the mean average precision (MAP) calculated over the precision values for each valid document found in the ranked documents retrieved by a system for each query. For example, if the first valid document is found in the third position of the ranking, the retrieval approach has a precision of 0.3. If the second is found in the fifth position, the precision is 0.2 and the MAP is 0.25. IAP and MAP are standard information retrieval measures [93] to evaluate ranked results from a retrieval system. The remaining rows present values of precision and recall in the first 5 (*@5*), 10 (*@10*) and 20 (*@20*) ranked documents for each system. Boldface entries indicate the best values for the three systems.

Values in Table 3.4 show a better performance for ip-CLAIRE on 4 of the 8 measures, while EM is better in the remaining 4, namely precision and recall in the first 5 and 10 ranked documents. This indicates that EM is actually better to recognize documents very close to the query, but for documents with fewer elements in common with the query, EM is not very precise. It can be better appreciated in Figure 3.4 where the interpolated precision values of ip-CLAIRE quickly overcome those of EM which is only better in 1 of the 11 recall points. This fact is also supported by the significant difference in the values of IAP and MAP between ip-CLAIRE and EM. For the 35 queries in the dataset, our approach took 42.23 seconds (1.2 seconds per query) to execute while for CLR took 1550.333 (44.29 seconds per query) showing an impressive enhancement in the computational time required to retrieve documents, a key issue in document retrieval. Both of these times include lattice construction.

3.3.4 Discussion on the capabilities of ip-CLAIRE

In this section we have presented a model for vector space retrieval based on FCA named ip-CLAIRE. While we have successfully shown that the VSM dynamics can be supported on a concept lattice of document descriptions, a new challenge arises from this scenario: By using

convex regions represented by interval patterns as formal concept intents we become unable to exploit some of the most important "affordances" of FCA for IR, namely enriching document corpora through external knowledge sources and providing relevance feedback based on query expansion and extension (see Section 1.4, Chapter 1).

Concept lattices in Figures 2.7 (Section 2.4) and 3.3 (Section 3.3) refer both to the same problem, this is picking a new formal concept associated to the query concept. In the first scenario, we use a semantic similarity measure relying on an external knowledge source while in the second we use the Euclidean distance in a vector space relying on a better representation of documents. Actually, we would like to merge both notions, exploiting the fact that we can relate documents by their semantics, accounting for relations not explicit in document corpora. At the same time we would like to use a more compact and dynamic document representation such as the vector model that naturally allows measuring and rank documents w.r.t. a given query. Therefore, we need to be able to represent documents in a *heterogeneous space*, where their descriptions can be a mixture of semantic annotations and points in a vectorial space.

To the author's knowledge, ip-CLAIRE is the first FCA-based IR model supporting the dynamics of the VSM retrieval paradigm. This is the main contribution of this work, however the questions it arises are of far more relevance. While the interval pattern structures is a good support for the VSM, in order to extend our work to heterogeneous document descriptions we are in the need for a new formalism of pattern structures.

3.4 A model for heterogeneous indexing

The idea of accounting for semantics in the VSM or the probabilistic retrieval model is not new and actually, it has an important body of work supporting it. Examples of these techniques are the "latent semantic indexing" (LSI) model [51], the "probabilistic latent semantic indexing" (PLSI) model [70] and the "latent Dirichlet allocation" (LDA) model [14]. We will refer to them as the "Latent Variables models" or LV-models, although sometimes they have been referred to as "topic models" [133].

LV-models are a widely spread, cutting-edge and useful manner to index, cluster and retrieve documents [94]. They share the basic notion that the information in a document collection is "generated" by a reduced set of latent variables (LVs) hidden in data, i.e. terms in a given document are a manifestation of topics or LVs (e.g. in an article about "formal concept analysis", the terms "formal context" and "concept lattice" are expected to be mentioned).

Nevertheless, latent variables are abstractions. While they may represent topics, those topics lack a proper characterization. This in turn, makes their interpretation a difficult task. For example, in the case of latent semantic indexing (LSI) [43] (considered to be the seminal work in topic models), LVs are represented by eigenvectors of a document-term matrix. Eigenvectors or convex regions in the eigenvector space (usually called "clusters") can be hardly recognizable as being, for instance, the topic of "formal concept analysis". Usually, we can try to manually recognize the documents and terms within a cluster to provide it with a proper *label*, however this can be expensive and tedious. Moreover, LV-models do not allow the incorporation of external knowledge sources which could aid in the "cluster labelling" task [93, 135].

Given the capabilities of FCA for classification and the dual representation of formal concepts (through the extent/intent description), LVs' characterization can be achieved through the use of relational context analysis (RCA) as introduced in Section 3.2.3. Specifically, through the construction of a relational context family containing a context of document descriptions in the latent variable space (an interval pattern structure), a formal context for terms' annotations

extracted from Wordnet, and a relational context between documents and terms representing the binary relation "document *is annotated with* term". Accordingly, a key aspect of this work is to address the issue that relational scaling is not currently supported for pattern structures. Therefore, as concluded in the previous section, we require a new model that would allow us to index documents in a heterogeneous space, where part of their description is given by a convex region in the latent variable space and another part is given by a set of relational scaled attributes created through the application of RCA. For this reason, in this section we present the "heterogeneous pattern structures" model.

3.4.1 Inspiring problem - Latent Semantic Indexing

Latent variables characterization problem

As previously discussed, LV-models lack a proper characterization for the LVs found through its application. For instance, Latent Semantic Indexing (LSI) [43] (sometimes referred to as "Latent Semantic Analysis" or LSA), a technique commonly used in information retrieval (IR) for indexation, clustering and dimension reduction purposes, is based on the idea that within a document-term matrix (as the one shown in Table 3.1) there is a set of hidden "latent variables" (LVs) that explain the data which constitutes the matrix. Consequently, LSI describes a technique to uncover these LVs through a "lower-rank approximation" of the original document-term matrix using linear algebra methods (specifically, singular value decomposition (SVD) [136]). By this, documents can be described not as vectors of term frequencies, but as vectors of LV values in a reduced vectorial space. Latent variables are supposed to capture the "semantics" in the set of documents, nevertheless it is difficult to grasp this notion while documents are still described by vectors of numeric values. In the following, we provide a further description of the LSI process as described in [43].

Latent Semantic Indexing

Let us consider the values in the formal context in Table 3.1 as a matrix A of dimensions 9×12. LSI works through the SVD of matrix A and the consequent calculation of the reduced space of LVs as follows:

$$A_{(9\times12)} = U_{(9\times9)} \cdot \Sigma_{(9\times12)} \cdot V^T_{(12\times12)} \tag{3.4}$$

$$\tilde{A}_{(9\times12)} = U_{(9\times k)} \cdot \Sigma_{(k\times k)} \cdot V^T_{(k\times12)} \quad (\text{with } k \ll min(9,12)) \tag{3.5}$$

$$A \sim \tilde{A} \tag{3.6}$$

$$\tilde{A} \cdot \tilde{A}^T = U_{(9\times k)} \cdot \Sigma_{(k\times k)} \cdot V^T_{(k\times12)} \cdot V_{(12\times k)} \cdot \Sigma^T_{(k\times k)} \cdot U^T_{(k\times9)} \tag{3.7}$$

$$\tilde{A} \cdot \tilde{A}^T = (U_{(9\times k)} \cdot \Sigma_{(k\times k)}) \cdot (U_{(9\times k)} \cdot \Sigma_{(k\times k)})^T \tag{3.8}$$

Where $(A)^T$ denotes the "transpose" of matrix A; U, V are orthonormal matrices and Σ is a diagonal matrix of "singular values". We have on one side the lower-rank approximation (Equation 3.6) to a matrix of rank k which is ensured to be the best $k - rank$ matrix approximation by the Frobenius norm difference [136]. On the other hand, we have the dimensional reduction (Equation 3.8) using matrix $U_{(9\times k)} \cdot \Sigma_{(k\times k)}$ as the space of documents in k LVs. Table 3.5 shows this space for matrix A with $k = 2$. Furthermore, Figure 3.6 presents a graphical representation of documents as points in a plane where we can appreciate the presence of 2 document groups,

	k1	k2
g_1	0.118	-0.238
g_2	0.046	-0.271
g_3	0.014	-0.413
g_4	0.014	-0.368
g_5	0.008	-0.277
g_6	0.519	0.002
g_7	0.603	-0.017
g_8	0.469	0.02
g_9	0.588	0.092

Table 3.5: Documents in 2 LVs

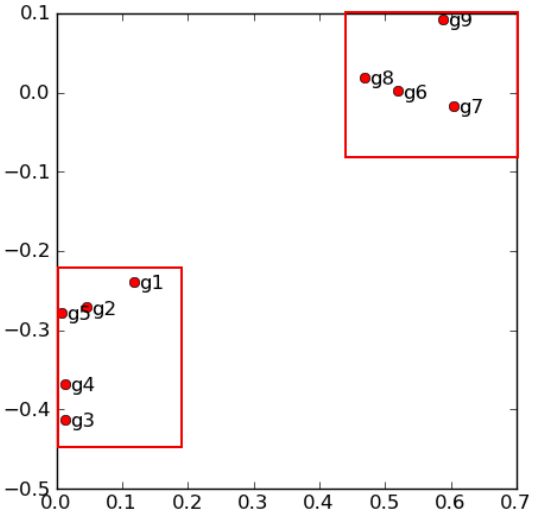


Table 3.6: Graphical representation of documents as points in a 2 dimensional LV space

usually called "clusters"[42]. In fact, one of the main uses of LSI is to provide a more compact representation of documents so that clusters are easier to find in the space of LVs. Incidentally, an interval pattern in this space represents a cluster (rectangles in Figure 3.6).

Problem statement

In Figure 3.6, while the clusters are easily distinguishable, it is not possible to say why they exist or what are their features. In order to characterize them we need to rely on their relations with terms. For example, we know that documents $g_6 - g_9$ share the term "arthroscopy"[43]. While this is not totally clear with documents $g_1 - g_5$ which do not share a common term, we can see that documents $g_1 - g_4$ share the term patient and g_2, g_3 and g_5 share the term user. Both terms are related through the annotation Person extracted from Wordnet (see Table 3.2b) which lead us to think that LVs can represent differences in this concern.

One way to automatically make these characterizations is through the use of the RCA framework where we can model documents and terms as objects, LV values as document descriptions, and Wordnet annotations as term attributes, while the document-term relation is given by aw in Table 3.2c. Nevertheless, as explained in the previous section, LSI generates document descriptions in the form of vectors of LV values, while clusters in the LV-space are better represented by interval pattern structures.

The main problem tackled in this work is how to enable the application of RCA in this kind of scenarios. We provide an adaptation of RCA which allows the relational scaling in pattern structures. We achieve this by the introduction of heterogeneous pattern structures described in the following section. A sub-goal of this work is to find out if domain knowledge can explain the existence and the "semantics" in LVs. We meet this sub-goal by the characterization of LVs through the proposed combination of RCA and pattern structures. Given that LVs define a

[42] Recall that we use the notion of "cluster" as a convex region in the LV space.
[43] We use the notation $g_6 - g_9$ to denote "all document between g_6 and g_9", including documents g_7 and g_8.

k-dimensional space (k being the number of LVs) where documents are organized, we formulate the following questions: Is it possible for us to find sub-regions in the space of LV values related to domain knowledge elements such as Wordnet annotations? And if so, how can we characterize these sub-regions?.

3.4.2 Adapting RCA for pattern structures

In this section we firstly describe the formal model description in which pattern structures are considered into a RCF. We show that the adaptation of the relational scaling operators induces a new space of heterogeneous object descriptions which we support in the framework of heterogeneous pattern structures. Following, we provide a full description of this novel pattern structures instance.

Formal Model

Consider the simple case when we have a single relation $\mathtt{r} \subseteq \mathtt{G_1} \times \mathtt{G_2}$ between two sets of objects, the domain of which is an object set in a pattern structure such as $\mathcal{K}_1 = (\mathtt{G_1}, (\mathtt{D}, \sqcap), \delta)$. The range of the relation is an object set inside a binary formal context $\mathcal{K}_2 = (\mathtt{G_2}, \mathtt{M}, \mathtt{I})$. Let us also define the relation as the set-valued function $\mathtt{r} : \mathtt{G_1} \rightarrow \wp(\mathtt{G_2})$ and let $\mathcal{L}_1 = \underline{\mathfrak{B}}(\mathcal{K}_1)$ and $\mathcal{L}_2 = \underline{\mathfrak{B}}(\mathcal{K}_2)$ be the pattern concept lattice and the concept lattice of $\mathcal{K}_1, \mathcal{K}_2$ respectively. Thus, we define the relational context family $(\mathbf{K}, \mathbf{R})$ where $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2\}$ and $\mathbf{R} = \{\mathtt{r}\}$. The usual RCA procedure induces iterations of formal context $\mathcal{K}_1$ through a "relational scaling" task using $\mathcal{L}_2$ ("target lattice" of $\mathtt{r}$), until the derived concept lattice $\mathcal{L}_1$ converges. For this reason, the scaling operators (universal, existential, etc.) are defined over a space of formal contexts into a space of formal contexts. This is the first complication in our model. Since in our setting $\mathcal{K}_1$ is a pattern structure and not a formal context, we cannot directly apply the scaling operators as defined in [126]. Thus, we move forward to redefine relational scaling operators which support pattern structures. To achieve this, let us define, for a relation $\mathtt{r}$, a function that assigns a set of relational attributes to a given object in the pattern structure depending on the type of relational scaling applied (universal, existential, etc.).

Definition 1. Let $\mathtt{r} \subseteq \mathtt{G_1} \times \mathtt{G_2}$ be a relation between two object sets where $\mathcal{L}_2$ is its target lattice composed by formal concepts $\mathtt{C}$. We define the potential set of all possible relational attributes $\mathtt{P_r}$ scaled from relation $\mathtt{r}$ as follows [44]:

$$\mathtt{P_r} = \{\mathtt{r} : \mathtt{C},\ \forall \mathtt{C} \in \mathcal{L}_\mathtt{j}\} \tag{3.9}$$

For $\mathtt{g} \in \mathtt{G_1}$, we also define two functions $\rho^{\exists}_{\mathtt{r}}, \rho^{\forall\exists}_{\mathtt{r}} : \mathtt{G_1} \rightarrow \wp(\mathtt{P_r})$ which assign a set of relational attributes to a given object using the 'existential quantifier operator ($\exists$)" and the "universal-existential quantifier operator ($\forall\exists$)" respectively.

$$\rho^{\exists}_{\mathtt{r}}(\mathtt{g}) = \{\mathtt{r} : \mathtt{C} \in \mathtt{P_r} \mid \mathtt{r}(\mathtt{g}) \cap \mathtt{extent}(\mathtt{C}) \neq \emptyset\} \tag{3.10}$$

$$\rho^{\forall\exists}_{\mathtt{r}}(\mathtt{g}) = \{\mathtt{r} : \mathtt{C} \in \mathtt{P_r} \mid \mathtt{r}(\mathtt{g}) \neq \emptyset, \mathtt{r}(\mathtt{g}) \subseteq \mathtt{extent}(\mathtt{C})\} \tag{3.11}$$

Hereafter we refer to $\rho^{\exists}_{\mathtt{r}}(g)$ or $\rho^{\forall\exists}_{\mathtt{r}}(g)$ as the "relations of $\mathtt{g}$".

[44] Normally, the relational attributes $\mathtt{r} : \mathtt{C}$ have the operator $\exists$ or $\forall\exists$ attached as a prefix indicating the scaling operation applied. In this work, we omit the prefixes in favour of generality. Nevertheless, the scaling function will remain indicated at each step.

Example 4. Let the following scenario be the running example for the remainder of this section. Consider a relational context family of two contexts $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2\}$ where $\mathcal{K}_1 = (\mathtt{G_1}, (\mathtt{D}, \sqcap), \delta)$ is the interval pattern structure of documents and their LV values shown in Table 3.5 and $\mathcal{K}_2$ is the formal context of terms and their Wordnet annotations shown in Table 3.2b. Consider as well the relation "document *annotated with* term" as shown in Table 3.2c such as $\mathbf{R} = \{\mathtt{r}\}$. From the initial lattice shown in Figure 3.2 we can construct the set of relational attributes $\mathtt{P_r} = \{\mathtt{aw} : \mathtt{C_i}\}$ where $\mathtt{i} \in [0, 7]$ (i.e. each $\mathtt{C_i}$ corresponds to one formal concept shown in the lattice[45]). Then, we have:

$$\begin{aligned}
\mathtt{r}(\mathtt{g_1}) &= \{\mathtt{patient}, \mathtt{laparoscopy}, \mathtt{scan}, \mathtt{complication}\} \\
\mathtt{extent(C1)} &= \{\mathtt{MRI}, \mathtt{scan}\} \\
\mathtt{r}(\mathtt{g_1}) \cap \mathtt{extent(C1)} &= \{\mathtt{scan}\} \neq \emptyset \implies \mathtt{aw} : \mathtt{C1} \in \rho_{\mathtt{r}}^{\exists}(\mathtt{g_1}) \\
\rho_{\mathtt{r}}^{\exists}(\mathtt{g_1}) &= \{\mathtt{aw} : \mathtt{C1}, \mathtt{aw} : \mathtt{C2}, \mathtt{aw} : \mathtt{C3}, \mathtt{aw} : \mathtt{C4}, \mathtt{aw} : \mathtt{C7}\}
\end{aligned}$$

Definition 2. Let $(\mathtt{G_1}, (\mathtt{D}, \sqcap), \delta)$ be a pattern structure for a set of objects $\mathtt{G_1}$ which are also associated with relational attributes in a set $\mathtt{P_r}$ through $\rho_{\mathtt{r}}^{\exists}$ or $\rho_{\mathtt{r}}^{\forall\exists}$. We define the scaled pattern structure $(\mathtt{G_1}, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta)$ with mappings $\Delta^{\exists}, \Delta^{\forall\exists} : \mathtt{G_1} \to \mathtt{H}$ as follows:

$$\mathtt{H} = \mathtt{D} \times \wp(\mathtt{P_r}) \tag{3.12}$$

$$\Delta^{\exists}(\mathtt{g}) = (\delta(\mathtt{g}), \rho_{\mathtt{r}}^{\exists}(\mathtt{g})) \tag{3.13}$$

$$\Delta^{\forall\exists}(\mathtt{g}) = (\delta(\mathtt{g}), \rho_{\mathtt{r}}^{\forall\exists}(\mathtt{g})) \tag{3.14}$$

Where $\mathtt{H}$ contains heterogeneous descriptions of objects in $\mathtt{G_1}$ combining both, a pattern $\delta(\mathtt{g}) \in \mathtt{D}$ and a set of relational attributes in $\mathtt{P_r}$.

Definition 3. Let $\mathtt{r}$ be a relation between two objects sets, then the existential scaling operator $(\mathtt{sc}_{\mathtt{r}}^{\exists})$ and the universal scaling operator $\mathtt{sc}_{\mathtt{r}}^{\forall\exists}$ for a pattern structure $\mathcal{K}_1$ are defined as:

$$\mathtt{sc}_{\mathtt{r}}^{\exists}(\mathcal{K}_1) = (\mathtt{G_1}, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists}) \qquad \mathtt{sc}_{\mathtt{r}}^{\forall\exists}(\mathcal{K}_1) = (\mathtt{G_1}, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\forall\exists}) \tag{3.15, 3.16}$$

As shown in Definitions 2 and 3, in order to apply the relational scaling operation to a pattern structure, it is necessary to define a new different pattern structure in which we can consider the original object description $\delta(\mathtt{g})$ and its relational attributes $\rho_{\mathtt{r}}^{\exists}(\mathtt{g})$ or $\rho_{\mathtt{r}}^{\forall\exists}(\mathtt{g})$. This combination of descriptions or "heterogeneous descriptions" $\mathtt{H}$ is a Cartesian product between the set of object descriptions and the powerset of $\mathtt{P_r}$ to which objects are mapped through $\Delta^{\exists} : \mathtt{G_1} \to \mathtt{H}$. We denominate this new pattern structure instance "heterogeneous pattern structures". In the following, we provide a complete description of its characteristics and capabilities.

Example 5. Table 3.7 shows a representation of the heterogeneous pattern structure of documents with LVs and relational attributes, where we can find an object description such as:

$$\begin{aligned}
\Delta^{\exists}(\mathtt{g_1}) &= (\delta(\mathtt{g_1}), \rho_{\mathtt{r}}^{\exists}(\mathtt{g_1})) \\
\delta(\mathtt{g_1}) &= \langle [0.118, 0.118], [-0.238, -0.238] \rangle \\
\rho_{\mathtt{r}}^{\exists}(\mathtt{g_1}) &= \{\mathtt{aw} : \mathtt{C1}, \mathtt{aw} : \mathtt{C2}, \mathtt{aw} : \mathtt{C3}, \mathtt{aw} : \mathtt{C4}, \mathtt{aw} : \mathtt{C7}\}
\end{aligned}$$

[45] aw stands for "annotated with". In the remainder of this section we will always work with the existential quantifier.

	D		P_r						
	k1	k2	aw : C1	aw : C2	aw : C3	aw : C4	aw : C5	aw : C6	aw : C7
g_1	0.118	-0.238	×	×	×	×			×
g_2	0.046	-0.271	×	×		×	×		
g_3	0.014	-0.413	×	×		×			×
g_4	0.014	-0.368	×	×		×			
g_5	0.008	-0.277				×	×		
g_6	0.519	0.002		×	×				×
g_7	0.603	-0.017		×	×				×
g_8	0.469	0.02		×	×				×
g_9	0.588	0.092		×	×				×

Table 3.7: Result of relational scaling in the example pattern structure represented in a hybrid formal context. We have removed the relational attribute aw : C0 usually assigned to every object.

Heterogeneous pattern structures

Definition 4. Let $H = D \times \wp(P_r)$ be a set of heterogeneous object descriptions, where $h_1 = (d_1, B_1)$ and $h_2 = (d_2, B_2)$ are two heterogeneous object descriptions with d_1, $d_2 \in D$, $B_1, B_2 \subseteq P_r$ and $h_1, h_2 \in H$ (the elements d and B are referred to as the "components" of h). We define the "similarity operator" $\sqcap_H$ between h_1 and h_2 as:

$$h_1 \sqcap_H h_2 = (d_1 \sqcap d_2, B_1 \cap B_2) \tag{3.17}$$

Example 6. The similarity operator applied to the object descriptions of g_1 and g_2 is:

$$\begin{aligned} \Delta^{\exists}(g_1) \sqcap_H \Delta^{\exists}(g_2) &= (\delta(g_1) \sqcap \delta(g_2), \rho_r^{\exists}(g_1) \cap \rho_r^{\exists}(g_2)) \\ \delta(g_1) \sqcap \delta(g_2) &= \langle [0.046, 0.118], [-0.271, -0.238] \rangle \\ \rho_r^{\exists}(g_1) \cap \rho_r^{\exists}(g_2) &= \{aw : C1, aw : C2, aw : C4\} \end{aligned}$$

Proposition 1. $(H, \sqsubseteq)$ with $\sqcap_H$ as described in Definition 4 is the direct product of the ordered sets $(D, \sqsubseteq)$ and $(\wp(P_r), \subseteq)$ and thus is an ordered set itself.

Proof 1. In order to prove that $(H, \sqsubseteq)$ is the direct product of $(D, \sqsubseteq)$ and $(\wp(P_r), \subseteq)$, we show that $h_1 \sqsubseteq h_2 :\iff d_1 \sqsubseteq d_2$ and $B_1 \subseteq B_2$ (as described in [63]).

$$\begin{aligned} h_1 \sqsubseteq h_2 &\iff h_1 \sqcap_H h_2 = h_1 && \text{Equation 1.13} && (3.18) \\ &\iff (d_1 \sqcap d_2, B_1 \cap B_2) = (d_1, B_1) && \text{Definition 4} && (3.19) \\ &\iff d_1 \sqcap d_2 = d_1 \text{ and } B_1 \cap B_2 = B_1 && && (3.20) \\ &\iff d_1 \sqsubseteq d_2 \text{ and } B_1 \subseteq B_2 && && (3.21) \end{aligned}$$

Because of Proposition 1, we would like to know how the heterogeneous pattern concept lattice is related to the concept lattices of its components, namely the pattern concept lattice

$(\mathtt{G}_1, (\mathtt{D}, \sqcap), \delta)$ and the concept lattice of the formal context of objects and their respective relational attributes $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ where the incidence relation $\mathtt{I}$ is defined in Equation 3.22. Regarding this, for the following definitions we introduce an alternative description for the standard FCA derivation operator $(\cdot)'$ in Equation 3.23 for a subset of objects $\mathtt{A} \in \mathtt{G}_1$ using the function $\rho^{\exists}_{\mathtt{r}}$.

$$\mathtt{I} = \bigcup_{\mathtt{g} \in \mathtt{G}_1} \{(\mathtt{g}, \mathtt{m}), \forall \mathtt{m} \in \rho^{\exists}_{\mathtt{r}}(\mathtt{g})\} \qquad \mathtt{A}' = \bigcap_{g \in \mathtt{A}} \rho^{\exists}_{\mathtt{r}}(\mathtt{g}) \qquad (3.22, 3.23)$$

Definition 5. The derivation operators $(\cdot)^{\diamond}$ in $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$ for an object set $\mathtt{A} \subseteq \mathtt{G}_1$ and a heterogeneous element $\mathtt{h} \in \mathtt{H}$ are defined as:

$$\mathtt{A}^{\diamond} = \prod_{\mathtt{g} \in \mathtt{A}} \Delta^{\exists}(\mathtt{g}) \qquad \mathtt{h}^{\diamond} = \{\mathtt{g} \in \mathtt{G}_1 \mid \mathtt{h} \sqsubseteq \Delta^{\exists}(\mathtt{g})\} \qquad (3.24, 3.25)$$

A heterogeneous pattern concept (hp-concept) is then defined as the pair $(\mathtt{A}, \mathtt{h})$ where $\mathtt{h}^{\diamond} = \mathtt{A}$ and $\mathtt{A}^{\diamond} = \mathtt{h}$.

Proposition 2. The derivation operator applied to a heterogeneous element $\mathtt{h} = (\mathtt{d}, \mathtt{B})$ is equal to the intersection of the derivation operator on its components:

$$(\mathtt{d}, \mathtt{B})^{\diamond} = \mathtt{d}^{\square} \cap \mathtt{B}' \qquad (3.26)$$

Proof 2. Let $\mathtt{g} \in \mathtt{h}^{\diamond}$, with $\mathtt{h} = (\mathtt{d}, \mathtt{B})$, by Equation 3.25 we have:

$$\mathtt{g} \in \mathtt{h}^{\diamond} \iff \mathtt{h} \sqsubseteq \Delta^{\exists}(\mathtt{g}) \iff \mathtt{d} \sqsubseteq \delta(\mathtt{g}) \text{ and } \mathtt{B} \subseteq \rho^{\exists}_{\mathtt{r}}(\mathtt{g}) \qquad \text{Proposition 1}$$

The right side of last formula shows two conditions. Using Equation 1.15, we have that the first condition yields $\mathtt{d} \sqsubseteq \delta(\mathtt{g}) \iff \mathtt{g} \in \mathtt{d}^{\square}$. As for the second condition, in Equation 3.22, we have that $(\mathtt{g}, \mathtt{m}) \in \mathtt{I}, \forall \mathtt{m} \in \rho^{\exists}_{\mathtt{r}}(\mathtt{g})$. Then, $\forall \mathtt{m} \in (\mathtt{B} \subseteq \rho^{\exists}_{\mathtt{r}}(\mathtt{g}))$ we have that $(\mathtt{g}, \mathtt{m}) \in \mathtt{I}$ and thus $\mathtt{g} \in \mathtt{B}'$. With this we have that:

$$\mathtt{g} \in (\mathtt{d}, \mathtt{B})^{\diamond} \iff \mathtt{g} \in \mathtt{d}^{\square} \text{ and } \mathtt{g} \in \mathtt{B}'$$
$$(\mathtt{d}, \mathtt{B})^{\diamond} = \mathtt{d}^{\square} \cap \mathtt{B}'$$

Proposition 3. The closure of a set of objects $\mathtt{A} \in \mathtt{G}_1$ (an extent) is equal to the intersection of its closures in each component.

$$\mathtt{A}^{\diamond\diamond} = \mathtt{A}^{\square\square} \cap \mathtt{A}'' \qquad (3.27)$$

Proof 3.

$$\mathtt{A}^{\diamond\diamond} = (\prod_{\mathtt{g} \in \mathtt{A}} \Delta^{\exists}(\mathtt{g}))^{\diamond} = (\prod_{\mathtt{g} \in \mathtt{A}} \delta(\mathtt{g}), \bigcap_{\mathtt{g} \in \mathtt{A}} \rho^{\exists}_{\mathtt{r}}(\mathtt{g}))^{\diamond} = (\mathtt{A}^{\square}, \mathtt{A}')^{\diamond} = \mathtt{A}^{\square\square} \cap \mathtt{A}''$$

From Proposition 3, we can see three different conditions for a heterogeneous extent $\mathtt{A}$, namely it can be closed in both of its components ($\mathtt{A}^{\diamond\diamond} = \mathtt{A}^{\square\square} = \mathtt{A}''$), in only one (either $\mathtt{A}^{\square\square} \subseteq \mathtt{A}''$ or $\mathtt{A}'' \subseteq \mathtt{A}^{\square\square}$), or in none ($\mathtt{A}^{\square\square} \nsubseteq \mathtt{A}''$ or $\mathtt{A}'' \nsubseteq \mathtt{A}^{\square\square}$). Further in is this section, we provide a full description for these kinds of extents. Nevertheless, Proposition 3 provides us with two ways to calculate the set of heterogeneous pattern concepts. Firstly, Equation 3.27 is a canonical test which can be used in standard FCA algorithms such as AddIntent [139]. Secondly, we can calculate the complete set of extents from both, the formal context and the pattern structure separately and intersect them to calculate each possible heterogeneous extent.

Extent $\mathtt{A}_i$	$(\mathtt{A}_i)^{\square\square}$	$(\mathtt{A}_i)''$	$\mathtt{A}^{\diamond\diamond} = \mathtt{A}^{\square\square} \cap \mathtt{A}''$	$(\mathtt{A}_i)^{\diamond}$
$\mathtt{A}_1 = \{\mathtt{g}_1, \mathtt{g}_3\}$	$\{\mathtt{g}_1 - \mathtt{g}_4\}$	$\{\mathtt{g}_1, \mathtt{g}_3\}$	$\{\mathtt{g}_1, \mathtt{g}_3\}$	-
$\mathtt{A}_2 = \{\mathtt{g}_5, \mathtt{g}_9\}$	$\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_5, \mathtt{g}_6, \mathtt{g}_8, \mathtt{g}_9\}$	$\mathtt{G}_1$	$\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_5, \mathtt{g}_6, \mathtt{g}_8, \mathtt{g}_9\}$	-
$\mathtt{A}_3 = \{\mathtt{g}_1, \mathtt{g}_6 - \mathtt{g}_9\}$	$\mathtt{A}_3$	$\mathtt{A}_3$	$\mathtt{A}_3$	$(\mathtt{A}_3^{\square}, \mathtt{A}_3')$
$\mathtt{A}_4 = \{\mathtt{g}_6, \mathtt{g}_7\}$	$\mathtt{A}_4$	$\mathtt{A}_3$	$\mathtt{A}_4$	$(\mathtt{A}_4^{\square}, \mathtt{A}_4')$
$\mathtt{A}_5 = \{\mathtt{g}_1, \mathtt{g}_3, \mathtt{g}_7\}$	$\{\mathtt{g}_1 - \mathtt{g}_4, \mathtt{g}_7\}$	$\{\mathtt{g}_1, \mathtt{g}_3, \mathtt{g}_6 - \mathtt{g}_9\}$	$\mathtt{A}_5$	$(\mathtt{A}_5^{\square}, \mathtt{A}_5')$

Table 3.8: Table showing different object sets under different closures. $\mathtt{A}_1$ is a proper extent of $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$ because it is closed under $(\cdot)^{\diamond}$ while $\mathtt{A}_2$ is not. $\mathtt{A}_3$ is the extent of a "pure hp-concept", while $\mathtt{A}_4$, of a "semi-pure hp-concept" (because $\mathtt{A}_4 \subseteq \mathtt{A}_3$). $\mathtt{A}_5$ is an extent of a "mixed hp-concept"

Example 7. Consider the object set $\mathtt{A}_1$ in Table 3.8. The closure in the fifth column shows that $\mathtt{A}_1 = \mathtt{A}_1^{\diamond\diamond}$ and thus it is a proper extent of $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$. This is not the case for $\mathtt{A}_2$.

Proposition 4. The closure of a heterogeneous description $\mathtt{h} \in \mathtt{H}$ is given by:

$$h^{\diamond\diamond} = (h^{\diamond\square}, h^{\diamond\prime}) \tag{3.28}$$

Proposition 4 can be demonstrated analogously to Proposition 3. We are interested in Proposition 4 because it allows us to easily calculate the heterogeneous intents as we show next.

Proposition 5. Let $\mathtt{A}_1$ be an extent in $(\mathtt{G}_1, (\mathtt{D}, \sqcap), \delta)$ and $\mathtt{A}_2$ be an extent in $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ where $\mathtt{A}_1 \subseteq \mathtt{A}_2$ and for any other extent $\mathtt{A}$ in $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ we have $\mathtt{A}_1 \subseteq \mathtt{A} \subseteq \mathtt{A}_2 \iff \mathtt{A}_2 = \mathtt{A}$, i.e. $\mathtt{A}_2$ is the cover of $\mathtt{A}_1$. Then for $\mathtt{h} = (\mathtt{A}_1^{\square}, \mathtt{A}_2')$, $\mathtt{h}$ is a heterogeneous intent and $(\mathtt{A}_1, \mathtt{h})$ is a hp-concept.

Proof 4. We show that $(\mathtt{A}_1^{\square}, \mathtt{A}_2')^{\diamond\diamond} = (\mathtt{A}_1^{\square}, \mathtt{A}_2')$

$$\begin{aligned}(\mathtt{A}_1^{\square}, \mathtt{A}_2')^{\diamond\diamond} &= (\mathtt{A}_1^{\square\square} \cap \mathtt{A}_2'')^{\diamond} = (\mathtt{A}_1 \cap \mathtt{A}_2)^{\diamond} = (\mathtt{A}_1)^{\diamond} \\ &= \bigsqcap_{\mathtt{g} \in \mathtt{A}_1} \Delta^{\exists}(\mathtt{g}) = (\bigsqcap_{\mathtt{g} \in \mathtt{A}_1} \delta(\mathtt{g}), \bigcap_{\mathtt{g} \in \mathtt{A}_1} \rho_{\mathtt{r}}^{\exists}(\mathtt{g})) \\ &= (\mathtt{A}_1^{\square}, \mathtt{A}_1') = (\mathtt{A}_1^{\square}, \mathtt{A}_2')\end{aligned}$$

The last step can be shown by the restrictions imposed to $\mathtt{A}_1$ and $\mathtt{A}_2$ as follows:
$\mathtt{A}_1 \subseteq \mathtt{A}_2 \implies \mathtt{A}_1 \subseteq \mathtt{A}_1'' \subseteq \mathtt{A}_2 \implies \mathtt{A}_1'' = \mathtt{A}_2 \implies \mathtt{A}_1' = \mathtt{A}_2'$

Similarly, it can be shown that when $\mathtt{A}_2 \subseteq \mathtt{A}_1$, the hp-concept $(\mathtt{A}_2, (\mathtt{A}_2^{\square}, \mathtt{A}_2'))$ exists. Proposition 5 shows that the extents in the pattern structure $(\mathtt{G}_1, (\mathtt{D}, \sqcap), \delta)$ and in $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ will be present in the lattice of hp-concepts. Nevertheless, these do not cover the whole set of hp-concepts in $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$.

As previously discussed, the set of hp-concepts (denoted as $\mathfrak{B}((\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists}))$) can be characterized as containing three types of extents, those that are closed under both components, those that are closed under one of its components and those that are an intersection of two different closed extents. We call these types "pure hp-concepts", "semi-pure hp-concepts" and "mixed hp-concepts" respectively.

Definition 6. Given a hp-concept $(\mathtt{A}, \mathtt{h}) \in (\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$ we say that:

$$(\mathtt{A}, \mathtt{h}) \text{ is "pure" iff } \mathtt{A}^{\square\square} = \mathtt{A}'' \tag{3.29}$$

$$(\mathtt{A}, \mathtt{h}) \text{ is "semi-pure" iff } \mathtt{A}^{\square\square} \subseteq \mathtt{A}'' \text{ or } \mathtt{A}'' \subseteq \mathtt{A}^{\square\square} \tag{3.30}$$

$$(\mathtt{A}, \mathtt{h}) \text{ is "mixed" iff } \mathtt{A}^{\square\square} \cap \mathtt{A}'' \neq \emptyset \text{ and } \mathtt{A}^{\square\square} \nsubseteq \mathtt{A}'' \text{ and } \mathtt{A}'' \nsubseteq \mathtt{A}^{\square\square} \tag{3.31}$$

Example 8. In Table 3.8, $\mathtt{A}_3$ is a pure hp-concept extent since it is closed in both components. $\mathtt{A}_4$ is a semi-pure hp-concept extent since it is closed in the pattern structure component but not in the relational attribute component. $\mathtt{A}_5$ is a mixed hp-concept as it is closed in the hp-lattice but not in either of its components.

In order to obtain the whole set of hp-concepts, it is not sufficient to calculate the sets of pattern concepts and formal concepts from its respective components and match them using Proposition 5. Doing so only provides us with the set of pure and semi-pure hp-concepts, while the set of mixed hp-concepts will be missing. In the following, we describe our method to compute the whole set of hp-concepts.

Calculating the hp-lattice

The heterogeneous pattern structure $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$ has been defined as a standard pattern structure and thus a standard algorithm to calculate pattern concept lattices can be used to obtain the hp-lattice. Some of these algorithms have been described and discussed in [79]. However, a much simpler manner to calculate the hp-lattice is through the use of a "scaled representation context".

A "representation context", as explained in [62], is a mechanism of complex data binarization. The pattern concepts of a pattern structure and the formal concepts of its derived representation context are in 1-1 correspondence and furthermore, their extents are the same [62, 85]. In the particular case of a heterogeneous pattern structure as described in this work, we use the representation context of the pattern structure component which is later "relationally scaled" in terms of traditional RCA (see Section 3.2.3).

Definition 7. Let $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$ be a heterogeneous pattern structure with components $(\mathtt{G}, \mathtt{M}, \mathtt{I})$ and $(\mathtt{G}_1, (\mathtt{D}, \sqcap), \delta)$. The "scaled representation context" is defined as $(\mathtt{G}_1, \mathtt{D} \cup \mathtt{P}_{\mathtt{r}}, \mathtt{J})$ where the incidence relation is:

$$(\mathtt{g}, \mathtt{x}) \in \mathtt{J} \iff \mathtt{x} \sqsubseteq \delta(\mathtt{g}) \text{ or } \mathtt{x} \in \rho^{\exists}_{\mathtt{r}}(\mathtt{g}); \forall \mathtt{g} \in \mathtt{G}_1 \text{ and } \mathtt{x} \in (\mathtt{D} \cup \mathtt{P}_{\mathtt{r}})$$

In other words, $(\mathtt{G}_1, \mathtt{D} \cup \mathtt{P}_{\mathtt{r}}, \mathtt{J})$ is the representation context of $(\mathtt{G}_1, (\mathtt{D}, \sqcap), \delta)$ plus the relational scaling of $(\mathtt{G}, \mathtt{M}, \mathtt{I})$. It can be shown that in fact, this "scaled representation context" is isomorphic to the representation context of $(\mathtt{G}_1, (\mathtt{H}, \sqcap_{\mathtt{H}}), \Delta^{\exists})$. For the running example, we constructed the scaled representation context as depicted in Table 3.9. In this context, patterns and relational attributes are treated equally, hence the attribute set $\mathtt{D} \cup \mathtt{P}_{\mathtt{r}}$. Patterns in $\mathtt{D}$ were filtered using a similarity threshold as described in [75], since the complete non-restricted pattern lattice contain a little more than 100 concepts. Incidentally, the filter by similarity applied to the calculation of $\mathtt{D}$ caused the hp-lattice derived from the context in Table 3.9 to contain only pure and semi-pure hp-concepts, i.e. their extents are either closed under $(\cdot)^{\square}$ or $(\cdot)'$ or both.

While there are some drawbacks w.r.t. the computational costs associated with the calculation of the formal concepts of the representation context, in this work we disregard them favouring the simplicity of the combined model.

3.4.3 Discussion on the heterogeneous pattern structures model

In Section 3.4.1 we proposed two questions that we answer now.
Is it possible for us to find sub-regions in the space of LV values related to domain knowledge elements? Indeed, we can. A hp-concept describes exactly this in its intent as a relation of an interval pattern and a set of annotations in the Wordnet taxonomy. Moreover,

	D																																P_r						
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	aw:C0	aw:C1	aw:C2	aw:C3	aw:C4	aw:C5	aw:C7
g1	×	×	×	×	×	×	×																										×	×	×	×	×		×
g2		×	×	×	×	×	×	×	×	×	×	×	×																				×	×	×		×	×	
g3				×	×					×	×			×	×	×																	×	×	×		×		×
g4			×	×	×	×			×	×	×	×			×	×	×	×															×	×	×		×		
g5					×	×	×				×	×	×			×		×	×														×				×	×	
g6																				×	×	×	×	×	×	×	×							×		×	×			×
g7																					×	×	×	×				×	×					×		×	×			×
g8																						×	×		×	×				×	×			×		×	×			×
g9																							×	×		×	×		×		×	×	×		×	×			×

Table 3.9: Scaled representation context for the running example. Patterns in D are represented by cardinals from 2 to 33 (number 1 was eliminated as it references the pattern concept $\top$).

these relations can be better described in the form of association rules [1]. Particularly, we are searching for those association rules with a premise in the space of latent variables and a consequence in the space of relational attributes, For example, we have the rule $6 \leftrightarrow \mathtt{aw:C4}$ which means that the latent variable region in the interval pattern numbered 6 implies the Wordnet concept Person as shown in Figure 3.5. Figure 3.5 presents a graphical representation for a set of association rules extracted from the running example. The map represents what can be called a "labelled hierarchical document clustering" [135] over the space of latent variables. In the map, the region marked as Activity is actually a union of two contiguous regions.

How can we characterize the relations among sub-regions in the space of LV values and domain knowledge elements? We have already described three types of hp-concepts, namely pure, semi-pure and mixed. In the following, we provide them with a characterization. Let us first introduce the Jaccard index [93] in terms of the hp-concept's extents and the extents of its components as follows ($|\cdot|$ represents set cardinality):

$$J(\mathtt{A}^{\square\square}, \mathtt{A}'') = \frac{|\mathtt{A}^{\square\square} \cap \mathtt{A}''|}{|\mathtt{A}^{\square\square} \cup \mathtt{A}''|} = \frac{|\mathtt{A}^{\diamond\diamond}|}{|\mathtt{A}^{\square\square} \cup \mathtt{A}''|}$$

Pure hp-concepts are interesting since they represent strong coherent relations between clusters in different spaces. Moreover, for any given pure hp-concept $(\mathtt{A}, \mathtt{h})$, the Jaccard index $J(\mathtt{A}^{\square\square}, \mathtt{A}'') = \mathtt{1}$. Consider for example, the pure hp-concept with extent $\mathtt{g_1} - \mathtt{g_5}$ (region 6) which represents a "closed" region in the latent variable space related to the topic People, i.e. outside this region, there are no documents related to People. We can also relate "pure hp-concepts" as describing *necessary and sufficient conditions* of a defined concept in the description logics framework (DL) [5]. In this case, documents in region 6 have the necessary and sufficient condition of being labelled with the annotation People.

A semi-pure hp-concept represents a directional coherence, i.e. either $\mathtt{A}^{\square\square} \subseteq \mathtt{A}''$ or $\mathtt{A}'' \subseteq \mathtt{A}^{\square\square}$. The Jaccard index is determined by $\frac{|\mathtt{A}^{\square\square}|}{|\mathtt{A}''|}$ in the first case or $\frac{|\mathtt{A}''|}{|\mathtt{A}^{\square\square}|}$ in the later. For example, the hp-concept with extent $\mathtt{g_6} - \mathtt{g_7}$ (region 22) contains documents related to Illness and Surgery, but it does not contain all of them (i.e. $\mathtt{g_1}$ is an exception). Thus, we can call a semi-pure hp-concept an "open" region in the latent variable space. In DL terms, semi-pure hp-concepts represent necessary conditions, i.e. being inside region 22 is a necessary but not sufficient condition for being labelled with the annotation Surgery.

Mixed hp-concepts represent a weak coherence of clusters. In general, their Jaccard index will be lower than the index of semi-pure hp-concepts.

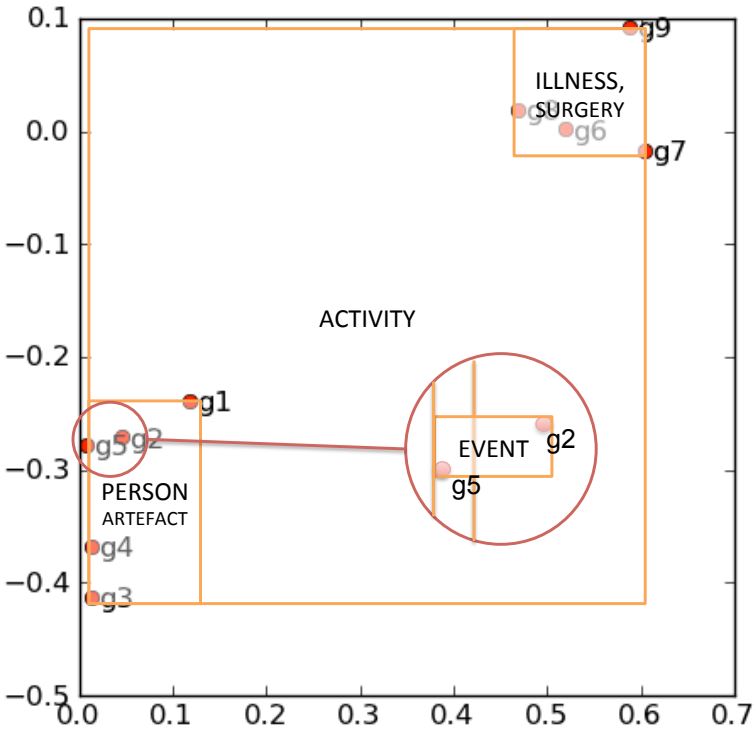


Figure 3.5: Labelled document clusters using association rules from the hp-lattice with magnification on documents g_2 and g_5

Finally, we can conclude that through the heterogeneous pattern structures model we are able to find useful relations among convex latent variable regions and domain knowledge which allows giving a proper characterization to the latent variable space, and hence, the latent variables themselves. This is possible due to the simultaneous representation of documents in the latent variable vectorial space and the set of relational attributes as hp-concepts.

In this work we have superficially described some connections with necessary and sufficient condition mining. Indeed, these connections may be further explored in the direction of possibility theory [46]. Furthermore, the notion of mixed hp-concepts, left unexplored in this work, leads us to think that they may be useful for annotation and data correction purposes. Other application domains seem also to fit as heterogeneous pattern structures. For example, in image annotation, images are characterized as vectors of features which are then aligned with annotations in the Wordnet taxonomy.

The main contributions of this work are the proposition of a coherent combination of pattern structures and RCA, the resulting description of heterogeneous pattern structures and a characterization technique for latent variables in a LV-model.

3.5 Conclusions

In Section 1.4, Chapter 1 we argued that, while FCA-based IR approaches are numerous and varied in their application domains, they rely over a limited pool of ideas, mainly due to the

fact that they mostly work as implementations of the same retrieval paradigm, the Boolean IR model.

In this chapter we have introduced a FCA-based IR approach which implements the dynamics of the vector space model for document retrieval. In order to achieve this, we have framed our approach within the definitions of the pattern structures framework, a FCA extension designed for the analysis of complex object descriptions. Our approach, named ip-CLAIRE (Interval Pattern Concept Lattices for Information REtrieval), is based on the notion of finding convex regions in the vector space used to represent documents as vectors which we identify as clusters of documents. The "hierarchical expansion" (HE) strategy for query modification commonly used in the Boolean retrieval model implemented over FCA has been re-used in our approach with a different focus. Specifically, HE allows us to derive a natural document ranking from a concept lattice for a given user query.

We have discussed the fact that, while this approach is feasible and has a better performance w.r.t. standard FCA-based IR applications and the exact matching retrieval technique, the number of patterns derived in the vector space is too high to consider this approach adequate for real document corpora. Furthermore, by changing the description space of documents to vectors instead of sets of terms, we have set aside some important advantages of the use of FCA for IR applications, namely the enrichment of document descriptions through the external knowledge sources and the support of relevance feedback techniques.

Regarding these issues, we have introduced a new indexing model which enables us to relate documents using heterogeneous descriptions. The *heterogeneous pattern structures* model allows us to mix vectorial representations and semantic annotations in order to create a "labelled overlapping hierarchical clustering" of documents. Heterogeneous pattern concepts also offer interesting perspectives of research derived from the relations they establish among the different descriptions spaces.

A good question regarding heterogeneous pattern concepts regards the semantics associated to a given hp-concept (heterogeneous pattern concept) in our setting. As previously described, the interval pattern concept was identified as a "cluster" of documents within a convex region of the vector space of descriptions. A hp-concept represents a cluster of documents plus a "cluster of semantic tags". This idea is very interesting as it resembles the notion of "bicluster", a technique extensively used in bioinformatics and information retrieval for pattern recognition, classification and data analysis [91]. This idea will be explored in depth in the following chapter.

Chapter 4

Beyond indexing: Biclustering and its applications

Contents

4.1 Introduction . . . 79
4.2 Background . . . 80
4.2.1 Biclustering . . . 80
4.2.2 The links between FCA and biclustering . . . 82
4.2.3 Triadic Concept Analysis. . . . 82
4.3 Biclustering using FCA . . . 83
4.3.1 Biclustering using partitions . . . 83
4.3.2 Formalizations . . . 83
4.3.3 Partition Space and Partition Pattern Structures . . . 85
4.3.4 Experiments . . . 88
4.3.5 Discussion on Biclustering and FCA . . . 89
4.4 FCA and Biclustering: Two additional models . . . 90
4.4.1 Scaled Partition Pattern Structures . . . 91
4.4.2 Interval Pattern Structure Approach . . . 92
4.4.3 Triadic Concept Analysis Approach . . . 93
4.4.4 Experiments . . . 94
4.4.5 Discussion . . . 96
4.5 Applications . . . 96
4.5.1 Recommender Systems . . . 96
4.5.2 Functional Dependencies . . . 101
4.6 Conclusions . . . 107

4.1 Introduction

Heterogeneous pattern structures (HPS) were introduced as a model for indexing documents w.r.t. several and distinct descriptions spaces (distinct in the nature of the data type, e.g. Boolean and numeric data). HPS is a model candidate for combining both, the Boolean and the

vector space retrieval models since it allows indexing documents using both paradigms simultaneously.

An interesting observation is that, while a document cluster gets indexed by a set of terms and a convex region, we could inversely consider that the set of terms gets indexed by the cluster of documents and the convex region. Traditionally, terms are used as elements for document clustering. However, term clustering has been proposed before in order to generate taxonomies for studying epistemic communities [125], ontologies for knowledge representation purposes [11], or bodies of knowledge for documenting digital libraries [33].

This dual view of "documents clustered by terms" and "terms clustered by documents" is organic in the FCA framework but, even more important, it constitutes a *bicluster*. Different from standard clustering where objects are compared and grouped together based on the full description space, biclustering generates groups of objects based on a subset of their attributes, values or conditions. Thus, biclusters are able to represent object relations in a local scale instead of the global representation given by an object cluster [91]. Biclustering has become a fundamental tool for bioinformatics and gene expression analysis [61], information retrieval [44] and more recently, for functional dependency mining [9].

In this chapter we move forward indexing towards data mining. We present a model for biclustering using FCA and the pattern structures framework. Moreover, we present an evaluation of this model comparing its performance w.r.t. a state-of-the-art biclustering technique. Next, we present a study on the relations of biclustering and FCA by introducing two additional models in which formal concepts and concept lattices can be used to generate and mine biclusters from a numerical data table. Finally, we present two applications of the aforementioned model, namely an application for film recommendation, and an application for functional dependency mining.

4.2 Background

4.2.1 Biclustering

A numerical data-table is a matrix A where A_{ij} indicates the value of an object $\mathtt{g}_i \in \mathtt{G}$ w.r.t. the attribute $\mathtt{m}_j \in \mathtt{M}$ with $i \in [1..|\mathtt{G}|]$ and $j \in [1..|\mathtt{M}|]$ ($|\cdot|$ represents set cardinality). A bicluster of A is a submatrix $\mathcal{B}$ where each value $\mathcal{B}_{ij}$ satisfies a given restriction. According to [61, 91], there are five different restrictions which we summarize in Table 4.1.

Constant values	$\mathcal{B}_{ij} = c$	Within the submatrix, all values are equal to a constant $c \in \mathbb{R}$ ($\mathbb{R}$ indicates real values).
Constant row values	$\mathcal{B}_{ij} = c + \alpha_i$	Within the submatrix, all values in a given row i are equal to a constant c and a row adjustment $\alpha_i \in \mathbb{R}$.
Constant column values	$\mathcal{B}_{ij} = c + \alpha_j$	Within the submatrix, all values in a given column j are equal to a constant c and a column adjustment $\alpha_j \in \mathbb{R}$.
Coherent values	$\mathcal{B}_{ij} = c + \alpha_i + \beta_j$	Within the submatrix, all values in a given column j are equal to a constant c, a row adjustment α_i and a column adjustment β_j. Instead of addition, the model can also consider multiplicative factors.
Coherent evolution		Values in the submatrix induce a linear order.

Table 4.1: Types of biclusters

Similar values instead of constant values

Restrictions in Table 4.1 define "biclusters with constant values", i.e. restrictions are based in the equality among different values in the submatrix. It is easy to observe that when noise is

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$	$\mathtt{m}_5$
$\mathtt{g}_1$	1	2	2	1	6
$\mathtt{g}_2$	2	1	1	0	6
$\mathtt{g}_3$	2	2	1	7	6
$\mathtt{g}_4$	8	9	2	6	7

Table 4.2: Bicluster with similar values ($\theta = 1$) - Example

	$\mathtt{m}_1$	$\mathtt{m}_3$
$\mathtt{g}_2$	2	1
$\mathtt{g}_3$	2	1

Table 4.3: Constant column bicluster - Example

present in a data-table (due to approximation issues or errors in data acquirement), it is difficult to expect that objects will have exactly the same value for a given attribute. Several approaches have tackled this issue in different ways, e.g. by the use of evaluation functions [112], equivalence relations [12, 104] and *tolerance relations* [77].

A tolerance relation is a reflexive and symmetric association between two objects. When it is also transitive, it is then denominated an equivalence relation [83]. In this work we will use a tolerance and an equivalence relation, namely one based on a given threshold ($\simeq_\theta$) and another based on equality ($=$), respectively. For a similarity threshold $\theta \in \mathbb{R}$ and any two numerical values $w_1, w_2 \in \mathbb{R}$, we say that $w_1 \simeq_\theta w_2 \iff |w_1 - w_2| \leq \theta$ (for numerical values, $|\cdot|$ refers to the absolute value) and it is read "w_1 is similar to w_2 w.r.t. a threshold θ".

We can adapt the first two value restrictions described in Table 4.1 to support tolerance relations as follows. Let $\mathcal{B}$ be a bicluster with n_1 rows and n_2 columns and let i, j, k, l be indexes such as $i, k \in [1, n_1]$ and $j, l \in [1, n2]$, then:

1. Similar values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kl}$
2. Similar row/column values:
   (a) Similar row values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{il}$
   (b) Similar column values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kj}$

Similar value biclusters requires that any value $\mathcal{B}_{ij}$ is similar to any other value $\mathcal{B}_{kl}$ in the bicluster. Similar rows requires that in row i all values are similar, i.e. any value $\mathcal{B}_{ij}$ has to be similar to any other value in the same column $\mathcal{B}_{il}$. In the same way, similar columns require that for column j, any value $\mathcal{B}_{ij}$ is similar to any other value in the same column $\mathcal{B}_{kj}$ It is worth noticing that when $\theta = 0$, similar value biclusters become constant value biclusters.

Example 9. With $\theta = 1$, Table 4.2 shows in its upper left corner a bicluster with similar values (dark grey). The upper right corner represents a similar column bicluster (light grey). Lower left corner considering $\{\mathtt{g}_3, \mathtt{g}_4\}$ and $\{\mathtt{m}_1, \mathtt{m}_2\}$ (not marked in the table) represents a similar row bicluster.

Hierarchy, overlapping and maximality

In the following, we adapt and combine some definitions described in [115, 77] for bicluster hierarchy, overlapping and maximality. Let $\mathcal{B} \in \mathbb{B}$ be a bicluster of the data-table A where $\mathbb{B}$ is the set of all biclusters. $\mathcal{B}$ has a unique correspondence with the pair of sets $(\mathtt{A}, \mathtt{B})$ where $\mathtt{A} \subseteq \mathtt{G}$ and $\mathtt{B} \subseteq \mathtt{M}$ such as for each $\mathtt{g}_i \in \mathtt{A}$ and for each $\mathtt{m}_j \in \mathtt{M}$, the cell $A_{ij} \in \mathcal{B}$. Let us define this correspondence with the mapping $\rho : \mathbb{B} \to \wp(G) \times \wp(M)$. We will define the order ($\leq$) between

two biclusters $\mathcal{B}^1$ and $\mathcal{B}^2$ equally to the definition of order $\leq_{\mathcal{K}}$ between two formal concepts as described in Equation 1.3, Section 1. Similarly, we can define overlapping between two biclusters if their mappings $\rho(\mathcal{B}^1) = (\mathtt{A}_1, \mathtt{B}_1)$, $\rho(\mathcal{B}^2) = (\mathtt{A}_2, \mathtt{B}_2)$ are incomparable w.r.t. $\leq$ as defined next.

- Hierarchy: $\mathcal{B}^1 \leq \mathcal{B}^2 \iff \rho(\mathcal{B}^1) \leq_{\mathcal{K}} \rho(\mathcal{B}^2)$.
- Overlapping: Two biclusters overlap iff $\rho(\mathcal{B}^1) \not\leq_{\mathcal{K}} \rho(\mathcal{B}^2)$ and $\rho(\mathcal{B}^1) \not\geq_{\mathcal{K}} \rho(\mathcal{B}^2)$ and $(\mathtt{A}_1 \cap \mathtt{A}_2 \neq \emptyset$ or $\mathtt{B}_1 \cap \mathtt{B}_2 \neq \emptyset)$.

Finally, a bicluster $\mathcal{B}$ is said to be maximal iff adding an object or (exclusive or) an attribute to the bicluster does not result in a bicluster [78], or if for its mapping $\rho(\mathcal{B}) = (\mathtt{A}, \mathtt{B})$ we have:

- $(\mathtt{A} \cup \{\mathtt{g}\}, \mathtt{B})$ does not have a pre-image $\rho^{-}1((\mathtt{A} \cup \{\mathtt{g}\}, \mathtt{B})) \in \mathbb{B}$.
- $(\mathtt{A}, \mathtt{B} \cup \{\mathtt{m}\})$ does not have a pre-image $\rho^{-}1((\mathtt{A}, \mathtt{B} \cup \{\mathtt{m}\})) \in \mathbb{B}$.

Remark. It should be noticed that in the formal definition, the similarity parameter θ is the same for all attributes. It is possible however to use a different parameter for each attribute without changing neither the problem definition nor the above described model. For real-world datasets, one can choose different similarity parameters $\theta_{\mathtt{m}}$ $(\forall \mathtt{m} \in \mathtt{M})$, but also can normalize/scale the attribute domains and use a single similarity parameter θ.

4.2.2 The links between FCA and biclustering

If we consider a given formal context to be a numerical data-table where crosses represent the value 1, and empty cells represent the value 0, then a formal concept is actually a maximal constant value bicluster (where $\mathcal{B}_{ij} = 1$). Moreover, the concept lattice provides an overlapping hierarchy of a set of constant value biclusters [116] as described above, however incomplete as FCA can only deal with binary data, i.e. the biclusters where $\mathcal{B}_{ij} = 0$ are not present (of course, they can be easily found by using a nominal scaling in the formal context, i.e. for each attribute $\mathtt{m}$, create another called $\neg\mathtt{m}$).

Usually, in FCA there are several techniques that allow data encoding to binary values by means of scaling [63, 77]. Nevertheless, this process introduces more parameters to the process [12] and increases its complexity [79, 9].

4.2.3 Triadic Concept Analysis.

A triadic context is given by $(\mathtt{G}, \mathtt{M}, \mathtt{N}, \mathtt{Y})$ where $\mathtt{G}$, $\mathtt{M}$, and $\mathtt{N}$ are respectively called sets of objects, attributes and conditions, and $\mathtt{Y} \subseteq \mathtt{G} \times \mathtt{M} \times \mathtt{N}$. The fact $(\mathtt{g}, \mathtt{m}, \mathtt{n}) \in \mathtt{Y}$ is interpreted as the statement "Object $\mathtt{g}$ has the attribute $\mathtt{m}$ under condition $\mathtt{n}$". A (triadic) concept of $(\mathtt{G}, \mathtt{M}, \mathtt{N}, \mathtt{Y})$ is a triple $(\mathtt{A}_1, \mathtt{A}_2, \mathtt{A}_3)$ with $\mathtt{A}_1 \subseteq \mathtt{G}$, $\mathtt{A}_2 \subseteq \mathtt{M}$ and $\mathtt{A}_3 \subseteq \mathtt{N}$ where if Equations 4.1 and 4.2 hold, then Equation 4.3 must hold.

$$\mathtt{A}_1 \times \mathtt{A}_2 \times \mathtt{A}_3 \subseteq \mathtt{Y}, \mathtt{X}_1 \times \mathtt{X}_2 \times \mathtt{X}_3 \subseteq \mathtt{Y} \tag{4.1}$$

$$\mathtt{A}_1 \subseteq \mathtt{X}_1, \mathtt{A}_2 \subseteq \mathtt{X}_2 \text{ and } \mathtt{A}_3 \subseteq \mathtt{X}_3 \tag{4.2}$$

$$\mathtt{A}_1 = \mathtt{X}_1, \mathtt{A}_2 = \mathtt{X}_2 \text{ and } \mathtt{A}_3 = \mathtt{X}_3 \tag{4.3}$$

If $(\mathtt{G}, \mathtt{M}, \mathtt{N}, \mathtt{Y})$ is represented by a three dimensional table, Equation 4.1 means that a concept stands for a 3-dimensional rectangle full of crosses while Equations 4.2 and 4.3 characterize

component-wise maximality of concepts. For a triadic concept $(\mathtt{A}_1, \mathtt{A}_2, \mathtt{A}_3)$, $\mathtt{A}_1$ is called the extent, $\mathtt{A}_2$ the intent and $\mathtt{A}_3$ the modus. To derive triadic concepts, two pairs of derivation operators are defined. The reader can refer to [88] for their definitions which are not necessary for understanding of the present work.

4.3 Biclustering using FCA

Biclustering has many elements in common with Formal Concept Analysis (FCA) [63]. In FCA objects are grouped together by the attributes they share in what is called a formal concept. Furthermore, formal concepts are arranged in a hierarchical and overlapping structure denominated a concept lattice. Hence, a formal concept can be considered as a bicluster of objects and attributes representing relations in a local scale, while the lattice structure gives a description in the global scale.

FCA is not only analogous to biclustering, but has much to offer in terms of mining techniques and algorithms [83]. The concept lattice can also provide biclusters with an overlapping hierarchy which has been reported as an important feature for bicluster analysis [116]. Recently, some approaches considering the use of FCA algorithms to mine biclusters from a numerical data-table have been introduced showing good potential [78, 77]. Our approach is a novel technique for lattice-based biclustering using the pattern structure framework [62], an extension of FCA to deal with complex data. More specifically, we propose a technique for mining biclusters with similar row/column values, a specialization of biclustering focused on mining attributes with coherent variations, i.e. the difference between two attributes is the same for a group of objects [91]. We show that, by the use of partition pattern structures [9], we can find high quality maximal biclusters (w.r.t. the mean squared error).

4.3.1 Biclustering using partitions

Consider that Table 4.2 represents the users (rows) ratings of a set of films (columns). Intuitively, we would like to find users which give the same rating for a given film (constant column) while letting ratings be different across different films (non-constant row). Actually, a bicluster containing users with the same taste in cinema.

Now, let us consider as an example the film $\mathtt{m}_3$ (attribute) in Table 4.2. We can observe that ratings (values) for this film "break" the set of users (objects) in two subsets, namely $\{\mathtt{g}_1, \mathtt{g}_4\}$ (rated with value 2) and $\{\mathtt{g}_2, \mathtt{g}_3\}$ (rated with value 1). Using a second film, for example $\mathtt{m}_5$, we can observe how both films *"break"* the space of users. This generates subsets $\{\mathtt{g}_1\}, \{\mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_4\}$. In particular, we can see how the pair $(\{\mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{m}_3, \mathtt{m}_5\})$ corresponds to a "bicluster with constant columns" as depicted in Table 4.3.

Our approach for mining biclusters with similar row/column values is based on this "breaking" or "partitioning technique". More specifically, it relies on partitions patterns found in a numerical data-table through equivalence relations and FCA algorithms. The hierarchical and overlapping structure supporting the biclusters is just a consequence of the lattice-based modelling provided by FCA.

4.3.2 Formalizations

A partition $\mathtt{d} = \{p_i\}$ of a set $\mathtt{G}$ can be formalized as a collection of components $p_i \subseteq \mathtt{G}$ such as:

$$\bigcup_{p_i \in \mathtt{d}} p_i = \mathtt{G} \qquad p_i \cap p_j = \emptyset \; ; (p_i, p_j \in \mathtt{d}, i \neq j)$$

The space of all partitions is denoted as $\mathtt{D}$ [9]. A partition $\mathtt{d}_1$ is a refinement of partition $\mathtt{d}_2$ (or $\mathtt{d}_2$ is a coarsening of $\mathtt{d}_1$) iff $\forall\ p_i \in \mathtt{d}_1, \exists\ p_j \in \mathtt{d}_2,\ p_i \subseteq p_j$. A partition of $\mathtt{G}$ can be created from an equivalence relation $[\mathtt{g}]$. An equivalence relation is a reflexive, symmetric and transitive binary relation between elements in a set. For example, the equivalence relation $[\mathtt{g}_i]_{\mathtt{m}_j}$ of an object w.r.t. an attribute is defined as follows:

$$[\mathtt{g}_i]_{\mathtt{m}_j} = \{\mathtt{g}_k \in \mathtt{G} \mid A_{ij} = A_{kj}\} \tag{4.4}$$

This allows creating a partition mapping $\delta : \mathtt{M} \to \mathtt{D}$ which assigns an attribute with a partition over $\mathtt{G}$ such as:

$$\delta(\mathtt{m}_j) = \{[\mathtt{g}_i]_{m_j} \mid \mathtt{g}_i \in \mathtt{G}\} \tag{4.5}$$

Example 10. From Table 4.2 we have $[\mathtt{g}_1]_{\mathtt{m}_5} = [\mathtt{g}_2]_{\mathtt{m}_5} = [\mathtt{g}_3]_{\mathtt{m}_5} = \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}$, $[\mathtt{g}_4]_{m_5} = \{\mathtt{g}_4\}$, and $\delta(\mathtt{m}_5) = \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_4\}\}$.

Analogously, we can define the tolerance block of an object $\mathtt{g}_i$ w.r.t. an attribute m_j and a given threshold θ as follows:

$$[\mathtt{g}_i]^{\theta}_{\mathtt{m}_j} = \{\mathtt{g}_k \in \mathtt{G} \mid A_{ij} \simeq_\theta A_{kj}\} \tag{4.6}$$

Where $A_{ij} \simeq_\theta A_{kj} \iff |A_{ij} - A_{kj}| \leq \theta$ is a tolerance relation as defined in Section 4.2.1.

Example 11. From Table 4.2 we have $[\mathtt{g}_1]^{\theta=1}_{\mathtt{m}_4} = [\mathtt{g}_2]^{\theta=1}_{\mathtt{m}_4} = \{\mathtt{g}_1, \mathtt{g}_2\}$, $[\mathtt{g}_3]^{\theta=1}_{m_4} = [\mathtt{g}_4]^{\theta=1}_{\mathtt{m}_4} = \{\mathtt{g}_3, \mathtt{g}_4\}$ and $\delta(\mathtt{m}_4) = \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}$.

Notice that when $\theta = 0$, Equations 4.4 and 4.6 are equivalent and thus, tolerance relations generalizes our approach for constant and similar value biclusters.

It is worth considering that our approach, as well as the partition pattern structures formalism proposed in [9] and introduced later in this section, are independent to the use of partitions or tolerance blocks. This election, however, has an impact in the application, namely the former allows us to mine constant biclusters while the latter, similar value biclusters. In the remainder of this chapter we will interchange both notions constantly for the sake of argument. The use of tolerance blocks can be inferred by the introduction of a θ value.

As described in the beginning of this section, using partition operations (searching for coincidences) we can discover biclusters with constant column values (or similar column values, using tolerance relations). In the following, we define these operations in the space of partitions $\mathtt{D}$. We show that $\mathtt{D}$ is an ordered space. We also show that the space of attributes $\mathtt{M}$ and the space of partitions $\mathtt{D}$ are related through a Galois connection [83, 63] that can be exploited in order to mine all possible bicluster pairs with constant column values from a numerical data-table reusing the algorithmic machinery of FCA.

4.3.3 Partition Space and Partition Pattern Structures

The space of all partitions $\mathtt{D}$ is a complete lattice [9, 104] where for two elements $\mathtt{d}_1$, $\mathtt{d}_2 \in D$ the meet and join are defined by Equations 4.7 and 4.8 and the order between two partitions is determined by Equation 4.9.

$$\mathtt{d}_1 \sqcap \mathtt{d}_2 = \bigcup_{p_i \in \mathtt{d}_1,\, p_k \in \mathtt{d}_2} (p_i \cap p_j) \tag{4.7}$$

$$\mathtt{d}_1 \sqcup \mathtt{d}_2 = \Big(\bigcup_{\substack{p_i \in \mathtt{d}_1,\, p_k \in \mathtt{d}_2 \\ p_i \cap p_j \neq \emptyset}} p_i \cup p_j \Big)^{+} \tag{4.8}$$

$$\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1 \tag{4.9}$$

Where we use $(\cdot)^{+}$ to denote closure for $\mathtt{d} \subseteq \wp(\mathtt{G})$ with components $p_i \subseteq \mathtt{G}$ such as:

$$\mathtt{d}^{+} = \{p_i \in \mathtt{d} \mid \nexists p \in \mathtt{d}, p_i \subseteq p\}$$

Intuitively, this means that the closure only conserves the maximal components in $\mathtt{d}$ not included in any other component. The meet of two partitions corresponds to the coarsest common refinement of two partitions. As we will describe later, this is the operation that allows us to enumerate all the possible biclusters. The join, on the other hand, represents the finest coarsening of two partitions. We do not use it for practical purposes in this work. The order between partitions establishes a hierarchical structure in the space $\mathtt{D}$ where coarser partitions subsumes finer partitions.

Example 12. Consider $\theta = 1$ for values in Table 4.2 we have $\delta(\mathtt{m}_1) = \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_4\}\}$, $\delta(\mathtt{m}_4) = \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}$ and $\delta(\mathtt{m}_5) = \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}\}$. We can calculate the following operations:

$$\begin{aligned}
\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) &= (\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\} \cap \{\mathtt{g}_1, \mathtt{g}_2\}) \cup (\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\} \cap \{\mathtt{g}_3, \mathtt{g}_4\}) \\
\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) &= \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\} \\
\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) &= \delta(\mathtt{m}_4) \iff \delta(\mathtt{m}_4) \sqsubseteq \delta(\mathtt{m}_5) \\
\delta(\mathtt{m}_1) \sqcup \delta(\mathtt{m}_4) &= \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}, \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_4\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}^{+} \\
&= \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}
\end{aligned}$$

Using the partition mapping function δ defined in Equation 4.5 and the ordered space of partitions $\underline{\mathtt{D}} = (\mathtt{D}, \sqcap)$, we can define a *partition pattern structure* $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$ using the framework described in Section 1.2.6. Partition pattern structures are an instance of the pattern structure framework initially proposed to mine functional dependencies among attributes of a database [9] dealing with set partitions.

It is worth noticing that in the current setting we are using $\mathtt{M}$ as the first element of the pattern structure $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$. Indeed, this is done to maintain the semantics of our definitions, since we are generating biclusters of objects which maintain similar attribute expressions making necessary to create partitions in $\mathtt{G}$. This approach yields "similar column biclusters". To generate "similar row biclusters" we can define the analogous pattern structure $(\mathtt{G}, \underline{\mathtt{D}}, \delta)$, where $\underline{\mathtt{D}}$ is now the space of ordered partitions of $\mathtt{M}$ and $\delta : \mathtt{G} \to \mathtt{D}$. Both approaches are conceptually equivalent.

Similarly to standard FCA, we have that $(\mathtt{B}, \mathtt{d})$ is a partition pattern concept (pp-concept) when $\mathtt{B}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{B}$ and that for two pp-concepts $(\mathtt{B}_1, \mathtt{d}_1)$ and $(\mathtt{B}_2, \mathtt{d}_2)$, the order between

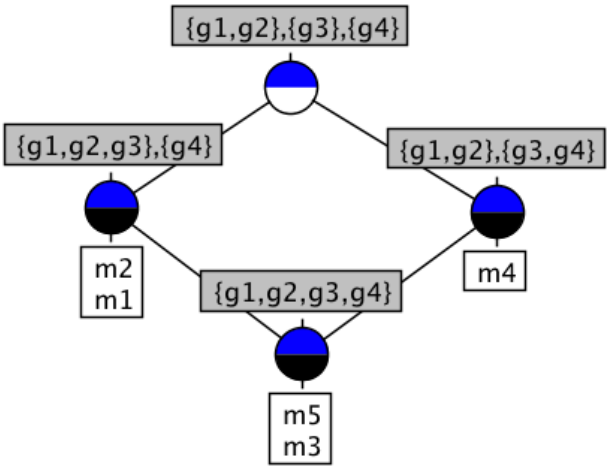


Figure 4.1: Partition pattern concept lattice

them is given by $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2) \iff (\mathtt{B}_1 \subseteq \mathtt{B}_2)$ or $(\mathtt{d}_2 \sqsubseteq \mathtt{d}_1)$. When $\mathtt{B}^{\square\square} = \mathtt{B}$, we ensure the maximality of pp-concept $(\mathtt{B}, \mathtt{d})$ and we call it a pp-concept. Pp-concepts determines biclusters as pairs $(p, \mathtt{B})$ where $p \in \mathtt{d}$ is a component of the partition pattern. It should be noticed that to keep consistency with previous notation, we write biclusters as pairs $(p, \mathtt{B})$ (p represent rows and $\mathtt{B}$ represent columns), while pp-concepts are written inversely $(\mathtt{B}, \mathtt{d})$ ($\mathtt{B}$ is the extent and $\mathtt{d}$ is the intent of $(\mathtt{B}, \mathtt{d})$).

Theorem 1. *Let $(\mathtt{B}, \mathtt{d})$ be a pp-concept, then for any partition component $p \in \mathtt{d}$ each pair $(p, \mathtt{B})$ corresponds to a similar column bicluster.*

The proof of this proposition is easy considering that each pair $(p, \mathtt{B})$ represents a submatrix the columns of which were selected using a tolerance relation, i.e. the values in the columns are similar w.r.t. $\simeq_\theta$.

We say that a bicluster $(p, \mathtt{B})$ is maximal iff adding an object to p or an attribute to $\mathtt{B}$ does not result in a bicluster, i.e. $(p \cup \{\mathtt{g}\}, \mathtt{B})$ and $(p, \mathtt{B} \cup \{\mathtt{m}\})$ are not biclusters.

Example 13. Figure 4.1 shows a partition pattern concept lattice (pp-lattice) created from Table 4.2 using $\theta = 1$. The pp-lattice contains 4 pp-concepts, but they correspond to 6 maximal similar column value biclusters listed in Table 4.4.

While pp-concepts are maximal (closed under $(\cdot)^\square$), biclusters corresponding to pairs $(p, \mathtt{B})$ are not always maximal. For instance, in Table 4.4 we have that bicluster 7 is not maximal because of bicluster 3. The same happens with bicluster 8 which is not maximal because of 4. This is due to the fact that pp-concepts are maximal w.r.t. the partitions and not w.r.t. the individual components of those partitions. Nevertheless, maximal biclusters are still easy to identify.

Theorem 2. *Let $(\mathtt{B}_1, \mathtt{d}_1), (\mathtt{B}_2, \mathtt{d}_2)$ be two pp-concepts such as $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2)$. Let $p \subseteq \mathtt{G}$ be a component of a partition. If $p \in \mathtt{d}_1$ and $p \notin \mathtt{d}_2$ then the bicluster corresponding to $(p, \mathtt{B}_1)$ is maximal.*

Proof 5. Given definitions in Equation 4.4 and the derivation operators $(\cdot)^\square$ defined for pattern structures, we have that for $(\mathtt{B}_1, \mathtt{d}_1)$ and for any $\mathtt{g}_i \in p$, the following is true:

$$p = \bigcap_{\mathtt{m}_j \in \mathtt{B}_1} \{\mathtt{g}_k \in \mathtt{G} \mid A_{ij} = A_{kj}\} \tag{4.10}$$

Consequently, for any other object $\mathtt{g}_h \in \mathtt{G}$, such as $\mathtt{g}_h \notin p$, we have $A_{ij} \neq A_{hj}$. Hence, the pair $(p + \{\mathtt{g}_h\}, \mathtt{B})$ cannot be a bicluster.

Let $\mathtt{B}_2 = \mathtt{B}_1 + \{\mathtt{m}_j\}$ for any $\mathtt{m}_j \in \mathtt{M}$, we show that $(p, \mathtt{B}_2)$ cannot be a cluster by contradiction. Let $(p, \mathtt{B}_2)$ be a bicluster. Then, there exists the pp-concept $(\mathtt{B}_2, \mathtt{B}_2^{\square})$ such as $p \in \mathtt{B}_2^{\square}$. If it does, then it is necessarily a direct super concept of $(\mathtt{B}_1, \mathtt{d}_1)$. However, this contradicts the definition $p \notin \mathtt{B}_2^{\square}$.

Intuitively, maximal biclusters can be found in a kind of "attribute concept" of the corresponding partition component (see Section 1.2, Chapter 1) (e.g. consider the maximal bicluster 6 in Table 4.4. It corresponds to the pp-concept labelled with $\mathtt{m}_4$ in Figure 4.1 and we can appreciate that $\{\mathtt{g}_3, \mathtt{g}_4\}$ is not present in the intent of its superconcept. The opposite happens with bicluster 7 in the concept labelled with $\{\mathtt{m}_1, \mathtt{m}_2\}$ in the lattice.). Indeed, partition pattern structures resembles a standard FCA process where the attribute set is multi-dimensional. In [9], it is shown how, through the use of transitive closures, the partition pattern structure is isomorphic to a binary formal context where the "scaling" procedure (encoding from many-valued data) generates a quadratic number of objects. We provide a deeper discussion on this issue in Section 4.4.

Mining biclusters

The calculation of the pp-concepts was implemented using AddIntent (the algorithm is described in [139]) for calculating a lattice of formal concepts. The algorithm was modified in order to obtain maximal biclusters from a numerical data-table. An important step to the process is the calculation of $\delta(\mathtt{m}_j)$ for a given $\mathtt{m}_j \in \mathtt{M}$, i.e. the initial object partition for a given column.

Calculating initial partitions: The equivalence relation described in Section 4.3.2 in the set of objects $\mathtt{G}$ given an attribute $\mathtt{m}_j$ is calculated using disjoint equivalence blocks. An equivalence block is an interval v over a range of values W (analogous to tolerance blocks described in [83]). For a given attribute $\mathtt{m}_j$, we define a set of disjoint equivalence blocks V_j where any two intervals $v_1, v_2 \in V_j$ have an empty intersection $v_1 \cap v_2 = \emptyset$. An object $\mathtt{g}_i$ belongs to the interval $v = [l, r[$ with $l, r \in W$ iff $l \leq A_{ij} < r$. Two objects $\mathtt{g}_i, \mathtt{g}_k$ are equivalent for V_j (given $\mathtt{m}_j$) iff they belong to the same interval $v \in V_j$. Each interval generates an equivalence class. This method has a complexity of $\mathcal{O}(\gamma|\mathtt{M}||\mathtt{G}|)$, where γ is the number of equivalence classes created. In general, equivalence blocks can be predefined by a user or calculated from W given γ, where $\mathtt{W}$ is the set of values taken by the attributes $\mathtt{M}$ for $\mathtt{G}$. In our experiments, we consider both, pre-defined equivalence blocks and using a γ value.

Bicluster	Elements	Maximal?
1	$(\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}, \{\mathtt{m}_3, \mathtt{m}_5\})$	✓
2	$(\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_5\})$	✓
3	$(\{\mathtt{g}_4\}, \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4, \mathtt{m}_5\})$	✓
4	$(\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4, \mathtt{m}_5\})$	✓
5	$(\{\mathtt{g}_3\}, \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4, \mathtt{m}_5\})$	✓
6	$(\{\mathtt{g}_3, \mathtt{g}_4\}, \{\mathtt{m}_3, \mathtt{m}_4, \mathtt{m}_5\})$	✓
7	$(\{\mathtt{g}_4\}, \{\mathtt{m}_1, \mathtt{m}_2\})$	✗
8	$(\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{m}_3, \mathtt{m}_4, \mathtt{m}_5\})$	✗

Table 4.4: Biclusters with similar values from pp-lattice

Breaking the lattice: As discussed at the end of Section 4.3.3, for a given pp-concept $(\mathtt{B}, \mathtt{d})$, not every pair $(p, \mathtt{B})$ with $p \in \mathtt{d}$ represents a maximal bicluster. Through the use of Proposition 2, we can easily discover biclusters within the pp-lattice while it is being calculated. Experimental data has shown that less than 20% of the pp-concepts within the pp-lattice actually hold a maximal bicluster. This has given room for an optimization of the AddIntent algorithm based on pruning the pp-lattice from pp-concepts not holding maximal biclusters. While this actually breaks the structure of the concept lattice, it keeps the order among the remaining pp-concepts, meaning that the output of AddIntent remains the same, while dramatically decreasing its computational time. This optimization has been included in the implementation of our biclustering algorithm.
Complexity: Aside the possible optimizations to the AddIntent algorithm, the fact remains that enumerating all possible biclusters from a data-table, from a large dataset, is computationally intractable [61, 91], since the number of biclusters grows exponentially w.r.t. the number of objects and attributes. For example, the complexity of the AddIntent algorithm is $\mathcal{O}(|\mathcal{L}| \cdot |\mathtt{G}|^2 \cdot |\mathtt{M}|)$ where $|\mathcal{L}|$ is the number of concepts in the lattice, which can grow up to $2^{|\mathtt{G}|}$ [139].

Methods to overcome this issue are still an open research problem in FCA, as well as in other disciplines. A popular technique in FCA is the use of minimal support for formal concept mining. This allows cutting from the lattice a set of formal concepts which are regarded as not representative and hence, not important for analysis purposes. The analogous of minimal support for biclustering is a minimal number of rows or columns required for a bicluster. This restriction has been used in several exhaustive bicluster enumeration algorithms to avoid exponential complexity [91]. In our approach, we set a minimal length for each equivalence class which we call σ. Each equivalence class such as $|[\mathtt{g}]_{\mathtt{m}_j}| \leq \sigma$ is ignored and not included the pp-concept intent.

4.3.4 Experiments

The first experiment shows the effect of the optimization discussed in the previous section. We used a subset of the dataset called MovieLens 100k[46] of film ratings containing 943 users and 50 films (out of a total of 1682) using $\sigma = 1$ (all bicluster sizes) and the predefined set of equivalence blocks $[1, 2][3, 3][4, 5]$. The dataset contains user ratings for films which range from 1 to 5. When information is not available, the matrix contains 0 which we disregard (we do not mine biclusters with columns equal to 0). The dataset contained 16532 similar column biclusters. The basic AddIntent algorithm processes a single object per iteration. Our experiment consists in inserting between AddIntent iterations a lattice pruning process while measuring the execution time. The dataset size was reduced to let $\sigma = 1$ and mine every possible bicluster. Results are shown in Figure 4.2. The solid horizontal line represents the execution time without optimization (30.5 seconds). While initially, the execution time doubles the non-optimized version (for a lattice prune at each AddIntent iteration), later the time quickly stabilizes around half the time the non-optimized version. The best time is found for 40 iterations (15 seconds).

The optimization affects the number of intent intersections performed by AddIntent. When the lattice is pruned, there are not as many intents to intersect as there were originally. However, pruning the lattice is an expensive task and adds overhead to the algorithm. The correct balance of this trade-off leads to dramatic improvements in the performance (twice in the experiments), however further experimentation in different numerical data-tables are needed to draw more conclusions regarding its setting.

The second experiment was performed over an example dataset provided with the system BicAt[47] containing 419 objects and 70 attributes. We measure the performance of our approach

[46] `http://grouplens.org/datasets/filmlens/`
[47] `http://www.tik.ee.ethz.ch/sop/bicat/`

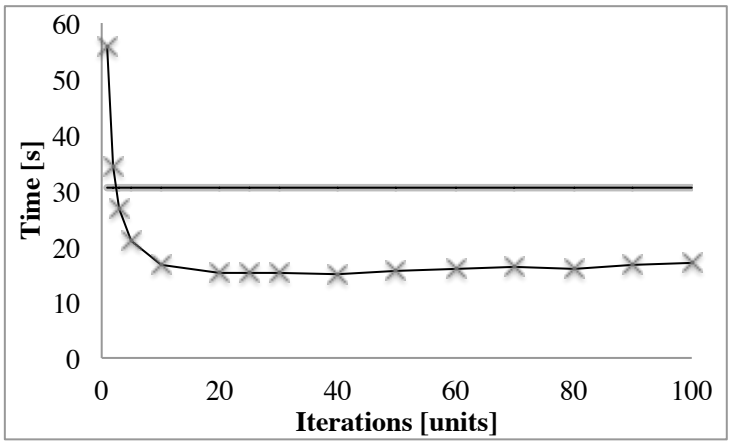


Figure 4.2: AddIntent Iterations per prune vs Execution time

mining similar row biclusters compared with Cheng and Church's algorithm (CC) [29]. CC tries to find a determined number of biclusters with a maximum threshold for the mean squared error δ. Results are shown in Table 4.5. Parameters for pp-lattice are number of equivalence blocks γ and minimal number of columns in the cluster σ. CC was executed as provided by BicAt and other parameters were left as system's default.

Results show a general better performance of our approach which is able to mine more than four million maximal biclusters from the dataset in less time than CC calculates only ten thousands. In terms of minimal squared error (MSE), our approach gets smaller scores which induces better quality biclusters. CC is able to find larger biclusters compared to our approach given the top-down strategy it implements. While larger biclusters can be found with our approach by decreasing the number of equivalent classes (γ), this is done at the cost of increasing the MSE as shown in Table 4.5. Compared to CC, our approach is better on finding many high quality and rather small biclusters inducing specialized associations among objects. CC is better at creating a global map of the entire data-table by finding larger biclusters.

4.3.5 Discussion on Biclustering and FCA

Biclustering techniques have usually been proposed for bioinformatics and gene expression analysis. A thorough description of these approaches can be found in [91] and a more recent one in [61]. FCA-based biclustering, on the other hand, is a relatively new approach for this problem.

	Time [s]	Biclusters [Kunits]	Parameters	MSE Max	Max Size [cells]
PPL	451	901	γ=20, σ=10	0.016	209
PPL	27	36	γ=10, σ=30	0.032	372
PPL	306	390	γ=10, σ=25	0.037	442
PPL	3,404	4,471	γ=10, σ=30	0.041	462
PPL	253	314	γ=5, σ=50	0.259	1,173
CC	418	1	$\delta = 0.5$	3.2	17,752
CC	416	1	$\delta = 0.3$	2.81	17,752
CC	4,018	10	$\delta = 0.5$	4.92	17,752

Table 4.5: Comparison between CC and pp-lattice bicluster algorithm

In [78], the authors present a technique based in interval pattern structures to mine similar value biclusters which was later revisited using Triadic Concept Analysis (TCA) in [77]. Our work shares many similarities with both of these approaches, however we focus on a different kind of biclustering, namely similar row/column value biclustering. Furthermore, we use a different framework and algorithms.

Lattice-based hierarchical clustering was also explored in [104] where the author proposes a technique analogous to single-linkage hierarchical clustering based in the Galois connection [63]. The author also introduces notions of "object" distance to support both, symbolic and numerical data. Our approaches extends from these notions to hierarchical biclustering and proposes an implementation not present in the aforementioned work.

It is worth mentioning that the type of biclustering tool that FCA can support has been catalogued as "exhaustive bicluster mining", this is finding all biclusters given a numerical data table. A similar approach was proposed in [12], in a system called NBS-miner (numerical bi-sets) introduced for constant value biclustering which also deals with tolerance relations using a threshold for "object" similarity. In the same lines, in [112], an approach based on range support patterns mining is proposed. In the latter, the authors propose a technique for exhaustive "similar" row biclustering using an evaluation function on the biclusters and the Apriori algorithm [87]. Both of these techniques present efficient ways for bicluster enumeration. Our approach differs in that it is able to find an overlapped hierarchical structure along with the biclusters.

To conclude, in this section we have insisted in the links between FCA and biclustering from the conceptual point of view which, in a nutshell, concerns the fact that both rely on the extraction of patterns by the dual relation between objects and attributes. We could say that biclustering differentiates in that it also involves coherence among values, but as we have described through this chapter, this is also true for the pattern structures extension of FCA. The next section is devoted to the description of the connections of FCA and biclustering, going as far as to introduce two additional settings regarding bicluster mining.

4.4 FCA and Biclustering: Two additional models

In this section we show that not only FCA provides a mathematical framework to characterize biclusters in several ways, but it also allows to compute them with existing and efficient algorithms. We describe two additional models for biclustering using different FCA extensions, namely interval pattern structures [79] (see Section 3.2.2, Chapter 3) and triadic concept analysis (TCA) [88] (see Section 4.2.2). Both of these models are conceptually different from each other and from the model based on partition pattern structures described in the previous section. Nevertheless, all these three models are equivalent in terms of results, i.e. they yield the same biclusters for a given numerical data table.

We will begin this section by introducing a "fourth method" for biclustering using standard FCA. In fact, this is just the result of a scaling procedure applied to a numerical data table modelled as a many-valued formal context and it is conceptually equivalent to the biclustering approach based on partition pattern structures defined in Section 4.3. By conceptually equivalent we mean that they yield isomorphic concept lattices.

Example. In order to better show the characteristics of each model, we will use a new numerical data table shown in Table 4.6: objects $\mathtt{G} = \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4, \mathtt{g}_5\}$ are represented by rows while attributes $\mathtt{M}$ $\{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4\}$ are represented by columns. $\mathtt{W} = \{0, 1, 2, 6, 7, 8, 9\}$ and we have for example $\mathtt{m}_4(\mathtt{g}_2) = 9$.

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$
$\mathtt{g}_1$	1	2	2	8
$\mathtt{g}_2$	2	1	2	9
$\mathtt{g}_3$	2	1	1	2
$\mathtt{g}_4$	1	0	7	6
$\mathtt{g}_5$	6	6	6	7

Table 4.6: A numerical data table

4.4.1 Scaled Partition Pattern Structures

The following model is derived from the partition pattern structure based biclustering algorithm described in Section 4.3. As we will show, both approaches are conceptually similar as they compute the same structures for biclustering mining, i.e. they derive isomorphic concept lattices. Actually, this model is based on a data transformation procedure described in [63] (page 92) slightly modified for our purposes.

Firstly, let us introduce a new notation for partitions that will be useful for future scenarios (more specifically, the notation used in functional dependencies). Let $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$ be a many-valued formal context modelling a numerical data table as the one shown in Table 4.6. We will define a partition over the set $\mathtt{G}$ as $\Pi_{\mathtt{m}} \subseteq \wp(G)$, where $c_i \in \Pi_m$ are called *components of the partition* with $c_i \cap c_j = \emptyset$ for all $c_i, c_j \in \Pi_{\mathtt{m}}$ s.t. $i \neq j$, and $\bigcup c_i = \mathtt{G}$, where $\mathtt{m} \in \mathtt{M}$ is the attribute that defines the equivalence relation corresponding to $\Pi_{\mathtt{m}}$. More formally:

$$\forall c_i \in \Pi_{\mathtt{m}}, \forall \mathtt{g}, \mathtt{h} \in c_i, \mathtt{m}(\mathtt{g}) = \mathtt{m}(\mathtt{h})$$

Thus, the partition pattern structure $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$ is defined with $\underline{\mathtt{D}}$ being the semi-lattice of partitions ordered by set inclusion and $\delta(\mathtt{m}) = \Pi_{\mathtt{m}}(\mathtt{G})$ is the mapping function assigning to each attribute its corresponding partition in $\mathtt{G}$.

It is noticeable that an equivalent formal context can be built. By equivalent, we mean that the concept lattices produced by both structures are isomorphic. Indeed, the formal context $(\mathtt{M}, \mathtt{G} \times \mathtt{G}, \mathtt{I})$ st. $(\mathtt{m}, (\mathtt{g}, \mathtt{h})) \in I \iff \mathtt{m}(\mathtt{g}) = \mathtt{m}(\mathtt{h})$ derives a concept lattice equivalent to the pattern concept lattice derived from $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$ [9], and thus it can be used in the same way to get maximal biclusters. This is of course, also applicable using tolerance blocks $\mathtt{m}(\mathtt{g}) \simeq_\theta \mathtt{m}(\mathtt{h})$. The proof can be done similarly as it is done in [9]. In our running example, such context is given in Table 4.7 for a tolerance of $\theta = 1$, and its associated concept lattice is given in Figure 4.3 (right). The lattice derived from the partition pattern concept using the same tolerance is shown 4.3 (right). It can be observed how these two lattices are isomorphic.

	(g_1, g_2)	(g_1, g_3)	(g_1, g_4)	(g_1, g_5)	(g_2, g_3)	(g_2, g_4)	(g_2, g_5)	(g_3, g_4)	(g_3, g_5)	(g_4, g_5)
m_1	×	×	×		×	×		×		
m_2	×	×			×	×		×		
m_3	×	×			×					×
m_4	×			×						×

Table 4.7: Formal context scaled from Table 4.6

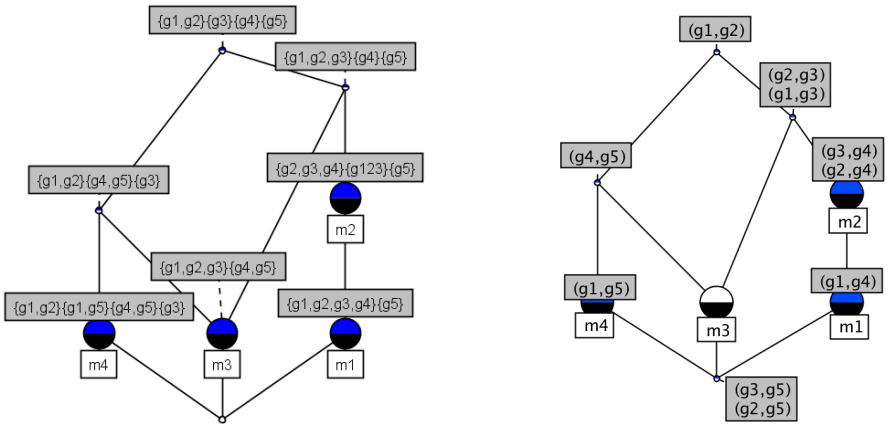


Figure 4.3: Pattern concept lattice on the left side, concept lattice of the right side

4.4.2 Interval Pattern Structure Approach

The basics of the interval pattern structure model are given in Section 3.2.2, Chapter 3. Let us recall that given a many-valued context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, in an interval pattern structure $(\mathtt{G}, \underline{\mathtt{D}}, \delta)$ objects are mapped to an interval vector in which each dimension follows a canonical order w.r.t. $\mathtt{M}$ (i.e. a dimension corresponds to the same attribute for all interval vector representations) and thus, the mapping function δ and similarity operator $\sqcap$ are defined with $\mathtt{g}, \mathtt{h} \in \mathtt{G}$ and $\mathtt{m}_j \in \mathtt{M}$ as follows:

$$\delta(\mathtt{g}) = \langle [\mathtt{m}_j(\mathtt{g}), \mathtt{m}_j(\mathtt{g})] \rangle,\ j \in [1, |\mathtt{M}|]$$
$$\delta(\mathtt{g}) \sqcap \delta(\mathtt{h}) = \langle [min(\mathtt{m}_j(\mathtt{g}), \mathtt{m}_j(\mathtt{h})), max(\mathtt{m}_j(\mathtt{g}), \mathtt{m}_j(\mathtt{h}))] \rangle,\ j \in [1, |\mathtt{M}|]$$

Examples. In Table 4.6, $(\{g_1, g_2, g_3\}, \langle [1,2], [1,2], [1,2], [2,9] \rangle)$ is a pattern concept:

$$\delta(\mathtt{g}_1) = \langle [1,1], [2,2], [2,2], [8,8] \rangle$$
$$\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}^{\square} = \delta(\mathtt{g}_1) \sqcap \delta(\mathtt{g}_2) \sqcap \delta(\mathtt{g}_3) = \langle [1,2], [1,2], [1,2], [2,9] \rangle$$
$$\langle [1,2], [1,2], [1,2], [8,9] \rangle \sqsubseteq \langle [1,2], [1,2], [1,2], [2,9] \rangle$$
$$\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}^{\square\square} = \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}$$

Interval pattern structures are particularly useful for mining similar value biclusters and thus, in the following definitions we will use the tolerance relation $\simeq_\theta$ instead of an equivalence relation, although as we have insistingly discussed, the use of one or the other does not affect the biclustering algorithm based in FCA.

Consider a function $attr : \mathtt{D} \rightarrow \mathtt{M}$ which returns for an interval pattern the set of attributes the interval of which is not larger than the θ parameter, for $\mathtt{d} = \langle [\mathtt{l}_j, \mathtt{r}_j] \rangle,\ j \in [1, |\mathtt{M}|]$: $attr(\mathtt{d}) = \{\mathtt{m}_j \in \mathtt{M} \mid \mathtt{l}_j \simeq_\theta \mathtt{r}_j\}$. Intuitively, *attr* "walks" through the interval vector measuring the width of each interval. If the width is less or equal than the θ parameter, then it returns the attribute corresponding to the current dimension of the vector (recall that there exists a canonical order within vectors w.r.t. the set $\mathtt{M}$). For example, for $\theta = 1$ and the ip-concept[48] $(\mathtt{A}_1, \mathtt{d}_1) = (\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \langle [1,2], [1,2], [1,2], [2,9] \rangle)$ extracted from Table 4.6, we have that the width of the first three dimensions corresponds to $2 - 1 = 1 \leq \theta$ while the width of the

[48] Interval pattern concept

fourth dimension is $9 - 2 = 7 \not\leq \theta$. Hence, $attr(\mathtt{d}_1) = \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3\}$. Similarly, for the ip-concept $(\mathtt{A}_2, \mathtt{d}_2) = (\{\mathtt{g}_1, \mathtt{g}_2\}, \langle [1,2], [1,2], [1,2], [8,9] \rangle)$ we have $attr(\mathtt{d}_2) = \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4\}$.

Proposition 6. Consider a numerical dataset $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$ as an interval pattern structure $(\mathtt{G}, \underline{\mathtt{D}}, \delta)$. For any maximal bicluster $\mathcal{B}$, there exists a pattern concept $(\mathtt{A}, \mathtt{d})$ such that for all cells $\rho(\mathcal{B}) = (\mathtt{A}, attr(\mathtt{d}))$.

Proof 6. The proof can be derived from the fact that in an ip-concept $(\mathtt{A}, \mathtt{d})$, we have that $\mathtt{d} = \mathtt{A}^{\square}$. Thus,

$$\begin{aligned} attr(\mathtt{d}) = attr(\mathtt{A}^{\square}) &= \{\mathtt{m}_j \in \mathtt{M} \mid max(\mathtt{m}_j(\mathtt{g}_i)) - min(\mathtt{m}_j(\mathtt{g}_i)) \leq \theta, \forall \mathtt{g}_i \in \mathtt{A}\} \\ &= \{\mathtt{m}_j \in \mathtt{M} \mid min(\mathtt{m}_j(\mathtt{g}_i)) \simeq_\theta max(\mathtt{m}_j(\mathtt{g}_i)), \forall \mathtt{g}_i \in \mathtt{A}\} \end{aligned}$$

It is clear that for any two pair of elements $\mathtt{g}_i, \mathtt{g}_k \in \mathtt{A}$, $\mathtt{m}_j(\mathtt{g}_i) \simeq_\theta \mathtt{m}_j(\mathtt{g}_k)$, $\forall \mathtt{m}_j \in \mathtt{M}$ or analogously, $A_{ij} \simeq A_{ik}$ by interpreting the many-valued context as a matrix A. This is the very definition of a bicluster and thus, we can establish a one to one relation between ip-concepts and biclusters. The maximality condition is ensured by the closure of the extent, i.e. $\mathtt{A}^{\square\square} = \mathtt{A}$.

4.4.3 Triadic Concept Analysis Approach

As previously discussed, triadic concept analysis (TCA) has been proposed for biclustering purposes in [77], particularly for mining constant value biclusters. In this section we present a different original result: any maximal bicluster of similar values is characterized as a triadic concept. The triadic context is derived from the numerical dataset by encoding the tolerance relation between values.

Proposition 7. Given a numerical dataset modelled as a many-valued context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, consider the derived triadic context given by $(\mathtt{M}, \mathtt{G}, \mathtt{G}, \mathtt{Y})$ s.t. $(\mathtt{m}_j, \mathtt{g}_i, \mathtt{g}_k) \in \mathtt{Y} \iff \mathtt{m}_j(\mathtt{g}_i) \simeq_\theta \mathtt{m}_j(\mathtt{g}_k)$. For any bicluster $\mathcal{B}$ such as $\rho(\mathcal{B}) = (\mathtt{A}, \mathtt{B})$, there exists a triadic concept $(\mathtt{B}, \mathtt{A}, \mathtt{A})$.

Proof 7. The proof to this proposition can be easily derived the definition of TCA provided in Section 4.2.2 and the proof of Proposition 6.

Example. Taking again $\theta = 1$, the triadic context derived from the numerical dataset from Table 4.6 is given in Table 4.8. The triadic concept $(\{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3\}, \{\mathtt{g}_1, \mathtt{g}_3, \mathtt{g}_2\}, \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\})$ corresponds to the maximal bicluster $(\{\mathtt{g}_1, \mathtt{g}_3, \mathtt{g}_2\}, \{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3\})$.

m_1	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×	×	
g_2	×	×	×	×	
g_3	×	×	×	×	
g_4	×	×	×	×	
g_5					×

m_2	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×		
g_2	×	×	×	×	
g_3	×	×	×	×	
g_4		×	×	×	
g_5					×

m_3	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×		
g_2	×	×	×		
g_3	×	×	×		
g_4				×	×
g_5				×	×

m_4	g_1	g_2	g_3	g_4	g_5
g_1	×	×			×
g_2	×	×			
g_3			×		
g_4				×	×
g_5	×			×	×

Table 4.8: Triadic context derived from Table 4.6 using $\simeq_1$

4.4.4 Experiments

We present experimental results with the different FCA methods introduced in the previous sections. We report preliminary results in two aspects: efficiency (running time) and compactness (number of concepts) to discuss the strengths and weaknesses of the different methods.

Data and experimental settings

The first dataset, "Diagnosis"[49], contains 120 objects with 8 attributes. The first attribute provides temperature information of a given patient with a range $[35.5, 41.5]$ (numerical). For this attribute we used $\theta = 0.1$ and then $\theta = 0.3$. The other 7 attributes are binary ($\theta = 0$). The second dataset, "dataSample_1.txt", is provided with the BiCat software[50]. It contains 420 objects and 70 numerical attributes with range $[-5.9, 6.7]$. We used $\theta = 0.05$ for all attributes. We provide results in Table 4.9 for the three different FCA methods discussed in this article, namely interval pattern structure (IPS), tolerance blocks/partition pattern structures (TBPS) and triadic concept analysis (TCA). The scaled partition pattern structures approach presented in Section 4.4.1 using a discretization technique is also evaluated. We propose a discussion on the computing of clarified contexts, given that it can dramatically reduce the size of the context while keeping the same concept lattice (FCA-CL). A context is clarified when there exists neither two objects with the same description, or two attributes shared by the same set of objects.

For the methods based on FCA and pattern structures (IPS, TBPS), we used a C++ version of the AddIntent algorithm [139][51]. No restrictions were imposed over the size of the biclusters. The TCA method was implemented using Data-Peeler [28]. All the experiments were performed using a Linux machine with Intel Xeon E7 running at 2.67GHz with 1TB of RAM.

[49] http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
[50] http://www.tik.ee.ethz.ch/sop/bicat/
[51] https://code.google.com/p/sephirot/

Results

Results in Table 4.9 show that for the Diagnosis dataset, the clarified context using standard FCA (FCA-CL) is the best of the five methods w.r.t. execution time while for the BicAt sample 1, the best is TCA. Times are expressed as the sum of the time required to create the input representation of the dataset for the corresponding technique and its execution. In the case of FCA and FCA-CL, the pre-processing can be as high as the time required for applying the AddIntent algorithm. However, for large datasets such as the BicAt example, these times can be ignored. It is also worth noticing that the pre-processing depends on the chosen θ value, hence for each different θ configuration, a new pre-processing task has to be executed. This is not the case for interval and partition pattern structures the pre-processing of which is linear w.r.t. the number of objects (it is actually, just a change of format). We can also appreciate a more compact representation of the biclusters by the use of partition pattern structures (TBPS) and its formal context versions (FCA and FCA-CL). While TBPS is the slowest of the five methods, it is also the cheapest one in terms of the use of machine resources, more specifically RAM. TCA is the more expensive method in terms of machine resources and data representation, however this yields results faster. Interval pattern structures are in the middle as a good trade-off of compactness and execution time.

For this initial experimentation we have not reported the number of maximal biclusters nor the bicluster extraction algorithms that can be implemented for each different technique, but only in the FCA techniques themselves. Regarding the number of maximal biclusters, this is the same for each technique since all of them are bicluster enumeration techniques, i.e. all possible biclusters are extracted. Hence, the difference among techniques is not given by the number of maximal biclusters extracted, but by the number of formal concepts found and their post-processing complexity to extract the maximal biclusters from them. In general, it is easy to observe that the post-processing of TCA is linear w.r.t. the number of triadic concepts found, while for IPS is linear w.r.t. the number of interval pattern concepts times the number of columns of the numerical dataset squared and for TBPS is linear w.r.t. the number of *super-sub* concept relations in the tolerance block pattern concept lattice. Nevertheless, different strategies for bicluster extraction can be implemented for each technique rendering the comparison unfair. For example, the optimization implemented for AddIntent described in Section 4.3.4 cuts in half the execution time by breaking the structure of the lattice for bicluster mining using partition pattern structures. Similar strategies for IPS and TCA could also be implemented but are still a matter of research.

	Diagnosis				BicAt sample 1	
	$\theta = 0.3$		$\theta = 0.1$		$\theta = 0.05$	
Technique	Time [s] Preproc + Exec.	#Concepts	Exec. Time [s] Preproc + Exec.	#Concepts	Exec. Time [s] Preproc + Exec.	#Concepts
FCA	0.11 + 0.335	**98**	0.11 + 0.291	**88**	2.3 + 2,220	**476,950**
FCA-CL	**0.11 + 0.02**	**98**	**0.11 + 0.011**	**88**	2.3 + 2,220	**476,950**
TCA	0.04 + 33.3	3,322	0.04 + 31.34	2,127	**3.17 + 360**	741,421
IPS	0.011 + 0.303	928	0.001 + 0.178	301	0.02 + 2,340	722,442
TBPS	0.011 + 1.76	**98**	0.001 + 0.411	**88**	0.02 + 5,340	**476,950**

Table 4.9: Number of concepts and execution times (pre-processing + AddIntent run)

4.4.5 Discussion

Biclustering is an important data analysis task that is used in several applications such as transcriptome analysis in biology and for the design of recommender systems. Biclustering methods produce a collection of local patterns that are easier to interpret than a global model represented by a set of plain or hierarchical clusters. There are several types of biclusters and corresponding algorithms, *ad hoc* most of the time. Our main contribution is to show how the *biclusters of similar values on columns* can be characterized or generated from formal concepts, pattern concepts and triadic concepts, deepening the links between biclustering and the formal concept analysis framework. This not only allows the efficient computation of biclusters, but also connects the biclustering problem with the algorithmic machinery of FCA.

4.5 Applications

In this section we report on two known applications of biclustering and present a model for which our approach may be useful. Namely, we present an application of biclustering for recommender systems and an application of the partition pattern structures model, the same used for biclustering, to mine functional dependencies from a relational database.

4.5.1 Recommender Systems

Recommender Systems (RSs) are a branch of information retrieval whose main goal is predicting which items from a catalogue would a user *consume* or *like* in the future given the characteristics of the item and the history of the user interactions with the system [123].

RSs have become increasingly popular since the 1999's "Netflix prize"[52], a competition for enhancing the precision of Netflix's trademark recommender system with a prize of a million dollars. Currently, a similar contest is being held in the context of the International Joint Conference on Artificial Intelligence (IJCAI 2015) [53] offering a prize of ten thousands dollars for predicting user returns to an e-commerce Website in an advertisement context.

Two main types of RSs have been described in [123], namely "content-based RSs" and "collaborative filtering RSs". In order to explain how both of these models work, let us first introduce some key concepts for RSs. Let $\mathtt{G}$ be a set of users and $\mathtt{M}$, a set of items, we can model a context for recommendation as a many-valued context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$ where $\mathtt{W}$ contains the range of possible ratings that users in $\mathtt{G}$ can give to items in $\mathtt{M}$ while $(\mathtt{g}, \mathtt{m}, \mathtt{w}) \in \mathtt{I}$ iff the user $\mathtt{g}$ has rated item $\mathtt{m}$ with value $\mathtt{w}$ or, what is the same $\mathtt{m}(\mathtt{g}) = \mathtt{w}$.

Furthermore, we can consider a couple of similarity functions $f_{\mathtt{g}} : \mathtt{G} \times \mathtt{G} \to [0,1]$, and $f_{\mathtt{m}} : \mathtt{M} \times \mathtt{M} \to [0,1]$ allowing two users being compared based on the items they have "liked" before, and to compare two items based on their characteristics (e.g. books compared by content, films compared by cast, etc.).

Given a user $\mathtt{g}$ and a set of items $\mathtt{B} \subseteq \mathtt{M}$ that he had liked in the past, "content-based RSs" recommends those items $\mathtt{n} \in \mathtt{M}$ s.t. $\sum_{m \in \mathtt{B}} f_{\mathtt{m}}(\mathtt{m}, \mathtt{n})$ is maximal. In simpler terms, it searches for all items in the collection which are similar to those that $\mathtt{g}$ has liked in the past.

The paradigm of "collaborative filtering RSs" is different and a little more complex. Given user $\mathtt{g}$ and the set of items he has rated in the past denoted as $\mathtt{g}'$ (observe that these are the items he has rated, but not necessarily *liked*). The system searches for all those users $\mathtt{h} \in \mathtt{G}$ such

[52] http://www.netflixprize.com
[53] http://ijcai-15.org/index.php/repeat-buyers-prediction-competition

that $f_{\mathtt{g}}(\mathtt{g}, \mathtt{h})$ is high. Then, the system recommends items $\mathtt{m} \in \mathtt{h}'$ such as $\mathtt{m} \notin \mathtt{g}'$ and $\mathtt{m}(\mathtt{h})$ is high. In simpler terms, it searches for all users similar to $\mathtt{g}$ and recommend him items they have liked and that $\mathtt{g}$ has not rated yet.

Any combination of these two paradigms is considered a "hybrid approach" which is usually regarded as the big third RSs model. One important aspect of RSs is the ability to provide context to results. This is of special importance in sensitive applications such as advertisement and news Web sites. For example, if a news article contains information about a plane crash, a content-based RSs may consider appropriate to recommend an air line Website given the keywords within the article. This recommendation, which sadly, is not an imaginary example at all, could seem very adequate from an algorithmic point of view, i.e. some number has been calculated which validates the recommendation. However, from a semantic perspective, this recommendation is wrong. Being able to indicate the application than in the *context* of a "plane crash", an air line Website is not an adequate recommendation is a key aspect to avoid this kind of issues.

Another important aspect of RSs is being able to explain a user why an item is being recommended. For example, recommendations of the type "we recommend item X because you *like* item Y" or "we recommend item X because user Z *like* it", allows the user to make the final decision and thus, in case that he does not *like* item X, not to blame the RS. This is specially useful in the case of films. For example, a user may like different genres of films and thus, different genres of films will be recommended to him. If he is in the "mood" of watching a film from a specific genre, then he can make the final evaluation from the recommendation explanations given to him.

In both these lines, as noted in Section 1.4, Chapter 1, FCA has been used as the backbone of recommender systems to support context in recommendation and explanations [27, 47, 73, 72, 130].

In here, we present some examples on how to use FCA-based biclustering to support a recommender system. We build over the previously discussed example of Section 4.3.1. Using biclusters for recommendation purposes is a relatively new approach and experimental evidence suggest a better performance than state-of-the-art RSs algorithm [4].

Consider the many-valued context in Table 4.10 (white cells)[54], where rows represent films, columns represent users, and cells represent a given rating for each film-user pair. Films and Users are also provided with an identifier. Grey cells represent a target user for recommendation $\mathtt{g}_t$, interrogation marks indicate that we do not know the value for that rating. Rating values are 1 for "dislike", 2 for "neutral" and three for "like". Now, let us consider a partition pattern structure $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$ where $\mathtt{M}$ is the set of films and $\underline{\mathtt{D}}$ represent the semi-lattice of ordered partition of the set of users $\mathtt{G}$.

[54] User icons obtained from `http://www.iconarchive.com/artist/jeanette-foshee.html`

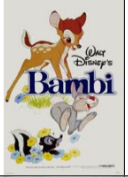
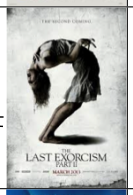
		g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_t
m_1		1	1	1	3	3	3	3	2	3	?
m_2		1	1	1	2	2	1	1	3	2	?
m_3		3	3	3	1	1	2	2	3	3	?
m_4		2	2	2	3	3	2	2	2	3	1
m_5		3	3	2	1	1	3	3	2	3	3

Table 4.10: A many-valued context for recommendation

Table 4.11 shows eleven maximal biclusters obtained from Table 4.10 (only white part of the table). These biclusters have a constant row restriction, i.e. users share the same ratings for a given set of films. For example, from bicluster 5 in Table 4.11 and consulting Table 4.10, we can obtain that each user has rated the group of films $\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_4$ with value 3,2,3 respectively. Biclusters in this context represent a group of users with the same taste regarding cinema.

Recommendation strategies

We will try to recommend a film for user $\mathtt{g}_t$ in the last grey column of Table 4.11. For this user we only have two ratings and thus, we will search for biclusters containing the films the user rated ($\mathtt{m}_4$ and $\mathtt{m}_5$). In Table 4.11 we can observe that they correspond to biclusters 1,2,3,6 and 9.

As we have pointed out, these biclusters give us a mean of agreement among users, a fact that can be exploited in several ways to provide recommendation. In the following we propose some possible strategies to use these biclusters for recommending purposes:

1. Naive strategy: Five biclusters can be used for the recommendation, however the target user $\mathtt{g}_t$ does not agree with all of them. For example, users in bicluster 2 have exactly the

opposite taste w.r.t the films within the bicluster than the target user, e.g. in the bicluster, films $\mathtt{m}_4$ and $\mathtt{m}_5$ are rated with values 3 and 1 while the target user has rated them with 1 and 3, respectively. Using this notion, it is possible to filter biclusters by agreement w.r.t. the target user and recommend items using those biclusters. For example, bicluster 1 seems to agree with the target user (e.g. 2,3 for the bicluster versus 1,3 for the target user in films $\mathtt{m}_4$ and $\mathtt{m}_5$). Users in bicluster 1 *like* film $\mathtt{m}_3$ and thus, we can recommend that film to the target user.

"We recommend film $\mathtt{m}_3$ because 2 users similar to $\mathtt{g}_5$ like it."

2. Average rating strategy: The previous strategy did not take into account that biclusters may disagree with each other. For example, bicluster 1 and 3 agree with the target user in the same way (2,3 for both biclusters versus 1,3 for the target user). However, film $\mathtt{m}_1$ is *disliked* in bicluster 1 while in bicluster 2 is *liked*. In the same way, film $\mathtt{m}_2$ is *liked* by bicluster 1 while it is *neutral* for bicluster 2. Instead of using single biclusters, it is possible to average ratings through several of them to obtain a better precision in the recommendation. For example, considering both biclusters, film $\mathtt{m}_1$ becomes *neutral*, while film $\mathtt{m}_2$ becomes between *neutral* and *liked* (with a rating 2.5) for the target user.

"We recommend film $\mathtt{m}_3$ because 4 users similar to $\mathtt{g}_5$ found it between *neutral* and *likeable*."

3. Disagreement strategy: The previous strategy disregarded bicluster 2 for being in perfect disagreement with the target user. In fact, this information can be also used for recommendation. If the target user is in perfect disagreement with the users of bicluster 2, then we could infer that he will *like* what users in bicluster 2 *dislike*. We can relate this rationale to that of analogical proportions [101]. Furthermore, disagreements can be measured and averaged through different biclusters. In this way, we could use the information in bicluster 2 to infer that the target user would *like* film $\mathtt{m}_3$ since users within bicluster 2 *dislike* it. In the same manner, we could infer that the target user would *dislike* film $\mathtt{m}_1$ since users in bicluster 2 *like* it. Averaging this information with that obtained in the previous strategy, we could infer that the target user *dislikes* film $\mathtt{m}_1$ and *likes* film $\mathtt{m}_3$.

4. Other considerations: It is hard to tell if it is preferable that a bicluster has more films or more users. As we know, these spaces are related through a Galois connection and thus, there is a trade-off between the number of films and the number of users within a given bicluster. Usually, the size of the bicluster is measured as a multiplication of both values, however it may be necessary to consider if it is more important that many users agree in some films, or that some users agree in many films. For example, "blockbusters"[55] will usually have a great part of the set of users *liking* them. This is actually how they became "blockbusters". Biclusters containing "blockbusters" may not be useful for recommendation purposes as users, disregarding their taste, will tend to like them. This is an example of how biclusters with many users may not be as important as biclusters with many films. In other contexts, the opposite may be true.

[55] Very popular films

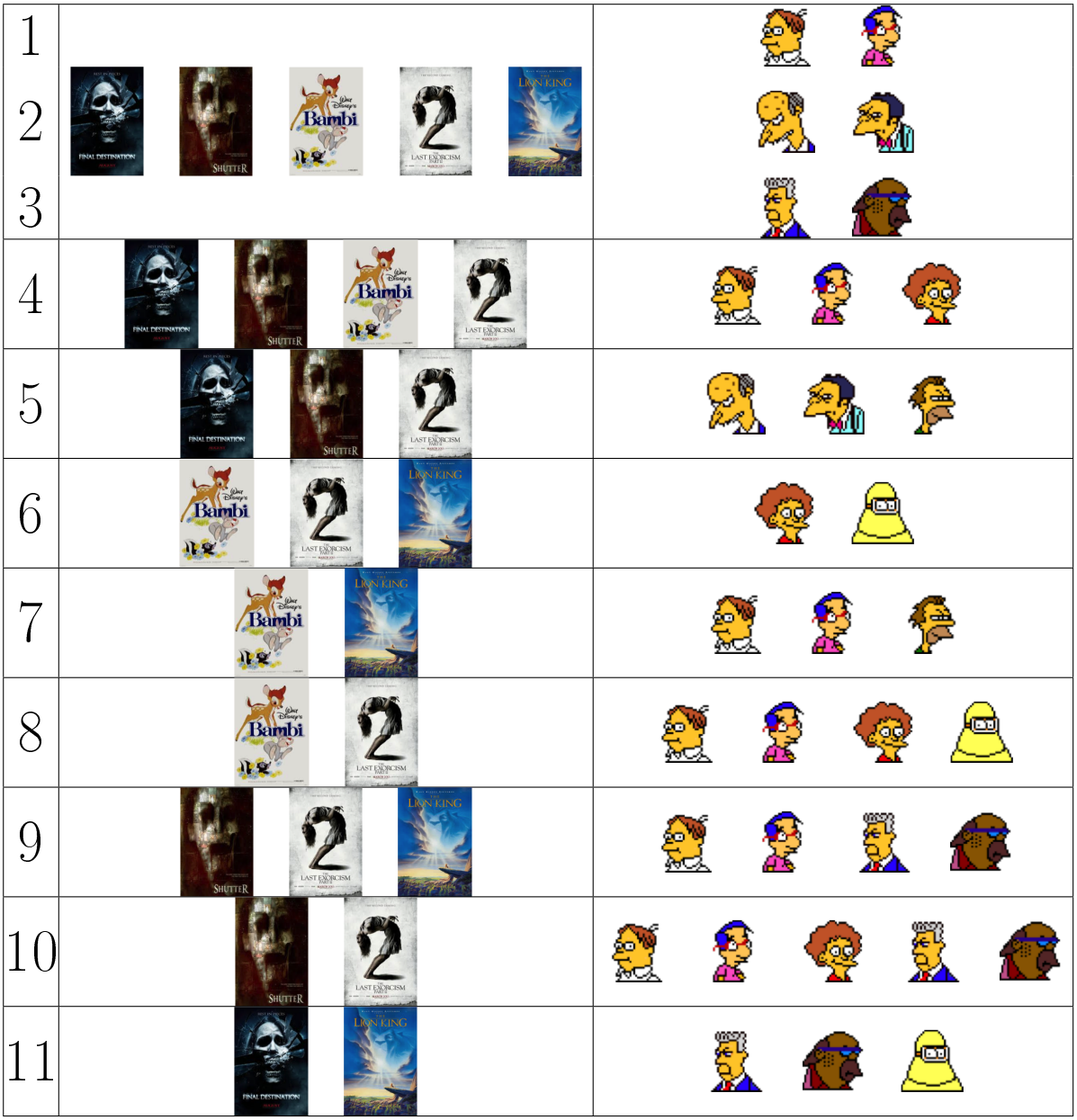


Table 4.11: Maximal biclusters obtained from the partition pattern structure modelled from Table 4.10. Each bicluster is characterized by an id, a group of films and a group of users. The first three biclusters share the same group of films

Considering heterogeneous information

Modern recommender systems rely in many information sources to improve the quality of their results. However, many times these information sources contain heterogeneous descriptions of the same objects. For example, as we have seen through our example, we can describe a film as a partition of the users that agree in the ratings that they have given to it. However, these films can also be described by their genres (sets of tags), their production dates (dates), locations (coordinate), etc.

Modelling each different information source is possible thanks to the heterogeneous pattern

structure framework described in Section 3.4. Through this model, we can describe a given film in several dimensions, e.g. partition of users, sets of genre tags, intervals of dates, etc. As a consequence, we are able to provide context to recommendations to avoid wrong results. For example, in Table 4.10 we have two different genres of films, namely horror films ($\mathtt{m}_1$,$\mathtt{m}_2$ and $\mathtt{m}_4$), and children films ($\mathtt{m}_3$ and $\mathtt{m}_5$). Bicluster 3 contains users that gave good ratings to both, horror films and children films. This bicluster represents users that, instead of being biased towards a genre of cinema, may rate films by their quality. Using this bicluster we could end up recommending a horror film to a user that historically has *liked* children films.

Through heterogeneous pattern structures, we can avoid this problem by characterizing films by their genre as well as the users that have liked them. Thus, the application can be provided with the context that horror films should not be recommended based on children films.

4.5.2 Functional Dependencies

In the relational database model, functional dependencies (FDs) are among the most popular types of dependencies since they indicate a functional relation between attribute sets [137], i.e. the values of an attribute set are determined by the values of another attribute set. These functional dependencies can be used to check the consistence of a database but also to guide the database design [92]. In some other cases, one should also deal with imprecision in the data, i.e. errors and uncertainty in real-world data.

Formally, a functional dependency can be defined between two attributes $\mathtt{m}_i, \mathtt{m}_j \in \mathtt{M}$ if we can establish a subsumption relation between the partitions they derive in the set of objects. In the following, we will re-use the formalization of partitions introduced in Section 4.4.1 and the many-valued context example in Table 4.7 which contain the initial formalization of partition pattern structures originally introduced in [9]. Let us recall that given a many-valued context $(\mathtt{G}, \mathtt{M}, \mathtt{W}, \mathtt{I})$, a partition $\Pi_{\mathtt{m}} \subseteq \wp(G)$ in the space of objects $\mathtt{G}$ (with components $c_k \in \Pi_{\mathtt{m}}$, $\bigcup c_k = \mathtt{G}$, and $c_k \cap c_l = \emptyset$ for all $c_k, c_l \in \Pi_{\mathtt{m}}$ with $k \neq l$) is derived using an equivalence relation determined by a given attribute $\mathtt{m} \in \mathtt{M}$ such that $\forall \mathtt{g}, \mathtt{h} \in c_k$, $\mathtt{m}(\mathtt{g}) = \mathtt{m}(\mathtt{h})$.

A functional dependency between two attribute sets $X, Y \subseteq \mathtt{M}$, denoted as $X \to Y$, is established iff $\Pi_X = \Pi_{X \cup Y}$. Thus, given a partition pattern structure $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$, if $X^{\square} = (X \cup Y)^{\square}$ then $X \to Y$. From this we can derive the following:

$$\begin{aligned} X \to Y &\iff X^{\square} = (X \cup Y)^{\square} \\ &\iff X^{\square} = X^{\square} \sqcap Y^{\square} \\ &\iff X^{\square} \sqsubseteq Y^{\square} \end{aligned}$$

In other words, given two attribute sets, if there is a subsumption relation between the partitions they derive in the set of objects, they are in a functional dependency relation. This allows us to derive trivial FDs from the set of partition pattern concepts extracted from $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$, this is, for two object concepts $\gamma(\mathtt{n}) \leq_{\mathcal{K}} \gamma(\mathtt{m})$ with $\mathtt{m}, \mathtt{n} \in \mathtt{M}$ then the FD $\mathtt{n} \to \mathtt{m}$ holds[56]. It is clear that FDs are in a one-to-one relation with the set of "object implications" which can be extracted from the concept lattice derived from $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$ as defined in Section 1.2.1, Chapter 1. Observe that the definition for "object implications" can be naturally derived from that of "attribute implications" by using extent relations instead of intent relations.

By extending our current model to use tolerance blocks instead of partitions (using tolerance instead of equivalence relations) we are able to support Similarity Dependencies (SDs) [8].

[56] Recall that $\mathtt{M}$ is the set of objects in the partition pattern structure $(\mathtt{M}, \underline{\mathtt{D}}, \delta)$

Example 14. Consider the many-valued context in Table 4.7 and a tolerance relation with $\theta = 1$:

$$\Pi_{\mathtt{m}_1} = \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}, \{\mathtt{g}_5\}$$
$$\Pi_{\mathtt{m}_2} = \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}, \{\mathtt{g}_5\}$$
$$\Pi_{\mathtt{m}_1} \sqcap \Pi_{\mathtt{m}_2} = \Pi_{\mathtt{m}_2} \implies \mathtt{m}_2 \to \mathtt{m}_1$$

Thus, there is a similarity dependency between $\mathtt{m}_2$ and $\mathtt{m}_1$. We can also observe this in the concept lattices of Figure 4.3, where $\gamma(\mathtt{m}_1) \leq_{\mathcal{K}} \gamma(\mathtt{m}_2)$.

As in the case of similar value biclusters explained in Section 4.3, the setting of SDs is a special case of our model and it depends on whether $\theta = 0$ (for FDs) or $\theta > 0$ (for SDs). In fact, the only real change in the formalization has to do with the components of the partition where for $\Pi_{\mathtt{m}}$, $c_k \cap c_l = \emptyset$ for all $c_k, c_l \in \Pi_{\mathtt{m}}$ does not necessarily hold. In the following, consider the definitions given for FDs applicable for SDs as well.

Mining FDs

Other than trivial FDs derived from relations between object concepts within the lattice, we can obtain FDs from each partition pattern concept using the notion of "intentional stability" provided in Section 1.2.2, Chapter 1. In the following, we provide a modified version to deal with pattern structures. Let $(\mathtt{A}, \mathtt{d})$ be a partition pattern concept, then its intentional stability is given by:

$$\sigma(\mathtt{A}, \mathtt{d}) = \frac{|\{\mathtt{C} \subseteq \mathtt{A} \mid \mathtt{C}^{\square} = \mathtt{A}^{\square}\}|}{2^{|\mathtt{A}|}}$$

Actually, for each $\mathtt{C}$ in the numerator of the previous formula we have that $\mathtt{C} \to \mathtt{A} \backslash \mathtt{C}$. This FD is verified by:

$$\begin{aligned} \mathtt{C}^{\square} &= (\mathtt{A} \backslash \mathtt{C} \cup \mathtt{C})^{\square} \\ &= (\mathtt{A})^{\square} \end{aligned}$$

Where the last line in the equation holds by the definition of intentional stability. The fact that the numerator of the stability formula is linked to the number of FDs associated to a given partition pattern concept is very useful since it gives us a tool to discriminate among the set of concepts, i.e. the highest the stability, the more FDs are associated to a given concept (actually, this should be corrected by a factor of $2^{|\mathtt{A}|}$). Current techniques of stability approximation [17] would allow for a quick assessment on the number of FDs in a given dataset and an enhancement in the performance of FD mining.

Example 15. Consider the meet of concepts labelled with $\mathtt{m}_2$ and $\mathtt{m}_3$ in concept lattices of Figure 4.3. The extent of this concept is $\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3$ and the numerator of its stability corresponds to the cardinality of the set with the following elements:

$$\{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3\}, \{\mathtt{m}_1, \mathtt{m}_3\}, \{\mathtt{m}_2, \mathtt{m}_3\}$$

Indeed, we have that $\{\mathtt{m}_1, \mathtt{m}_3\} \to \mathtt{m}_2$ and $\{\mathtt{m}_2, \mathtt{m}_3\} \to \mathtt{m}_1$ (let us obviate the trivial SD associated with the first set).

Experiments

In the following we present an experimental evaluation of the proposed model for SDs mining. Following the lines of the experiments presented for biclustering mining, we evaluate our approach using three different methods, namely using partition pattern structures, scaling and standard FCA, and scaling and standard FCA using clarified contexts (explained later in this section). We experimented on an Intel Xeon machine with 6 cores running at 2.27GHz and 32 GB of RAM machines, and all algorithms have been implemented in C++ and compiled with the $-O3$ optimization.

Dataset description and experimental settings: We experimented with 3 datasets from the UCI machine learning repository and 3 datasets from the JASA data archive, namely the *diagnosis*[57] *contraceptive*[58], *servo*[59], *caulkins* [60], *hughes-r* [61], and *pglw00* [62] datasets, described in Table 4.13.

Except for *caulkins* and *hughes-r*, all datasets were used with no modification. In the case of *caulkins*, some columns were ignored. Specifically, we did not consider the columns with redundant information about the weight of the entry (there where two columns indicating this weight, one with the original information and another with its correspondent gram representation. We used the gram representation and a column containing the value "gram" for every object. In the case of *hughes-r* we added four columns to encode the information of the first three columns. All additional columns are binary. Furthermore, the value -1 was used to indicate "empty value", meaning that this information should not be considered, i.e. for the pattern intents generated for the three first columns, objects with the value -1 are not present in any component. Changes to other datasets just considered the conversion of categorical entries represented with a string to a number.

The discretization procedure to turn a dataset into a formal context (i.e. a binary relation) is achieved via a simple script: for any two objects in `G`, it gives the attributes from `M` for which those objects agree, i.e. have similar values. Moreover, we consider also an operation called context clarification which avoids producing two different pairs of objects that agree for exactly the same attributes. It indeed has no impact on the concept lattice which holds the same dependencies [63]. It however significantly reduces the size of the formal context. As such, we produce both the non-clarified and clarified contexts for a given dataset. For processing the resulting formal contexts, we use an implementation of the *AddIntent*[63] algorithm [139] which allows building the concept lattice from which similarity dependencies can be characterized.

To proceed with pattern structures, each attribute $\mathtt{m} \in \mathtt{M}$ of the original dataset is described by tolerance blocks of objects from `G` which depends on the chosen similarity parameter $\theta_{\mathtt{m}}$. Thanks to the genericity of pattern structures, and to be fair in the algorithmic comparison of the two approaches, we modified the same *AddIntent* algorithm implementation to process a pattern structure. The modification consists in overriding the computation of description intersections, and the subsumption test for any two descriptions. Both operations are quadratic w.r.t number of original objects `G` in the worst case scenario. For the sake of efficiency, we use striped descriptions, i.e. we do not keep singletons in partition patterns, as in [90] and [7].

[57] `http://archive.ics.uci.edu/ml/datasets/Acute+Inflammations`
[58] `http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice`
[59] `http://archive.ics.uci.edu/ml/datasets/Servo`
[60] `http://lib.stat.cmu.edu/jasadata/caulkins-p`
[61] `http://lib.stat.cmu.edu/jasadata/hughes-r`
[62] `http://lib.stat.cmu.edu/jasadata/pglw00.zip`
[63] `https://code.google.com/p/sephirot/`

Finally, it should be realized that the difference between using tolerance relations or using partitions as attribute descriptions $\delta(m)$ (as presented in [7]) is that tolerance blocks are not restricted to have an empty intersection, i.e. they can overlap. This has a direct impact on the computing efficiency for calculating the concept lattice, as we will see later in this section. Parallelization using the OpenMP[64] library was used to calculate tolerance block intersections improving the efficiency of the lattice building algorithm.

Experimental results: We process each dataset as follows: (i) with the standard *AddIntent* algorithm we build the *concept lattice* of the derived formal context ; (ii) with a *modified AddIntent* algorithm we build the *pattern concept lattice* of the pattern structure. Table 4.13 reports the execution times for building those lattices. For non-clarified and clarified formal contexts, execution times report a sum of the binarization/clarification time and the execution of the AddIntent algorithm, respectively. For pattern structures, the execution times take into account the transformation of the numerical dataset into a pattern structure as well as its processing. Notice that not all datasets have the same number of categorical and numerical attributes. The similarity parameter is used only for numerical attributes, however for categorical attributes θ-values are always 0 (equivalence relation). Table 4.12 shows the different values taken by theta for all datasets. An important observation is that θ-values where selected arbitrarily with no regard for its actual meaning in the application domain of the dataset, but only considering computational purposes.

In all the chosen datasets (except for *pglw00*), processing the formal contexts is more efficient than processing the equivalent pattern structure. Formal context clarification takes the same time as the non-clarified formal context building: when a pair of objects $(\mathtt{g}_1, \mathtt{g}_2) \in (\mathtt{G} \times \mathtt{G})$ is generated, we check that if we had already generated another pair $(\mathtt{h}_1, \mathtt{h}_2) \in (\mathtt{G} \times \mathtt{G})$ such that $(\mathtt{g}_1, \mathtt{g}_2)' = (\mathtt{h}_1, \mathtt{h}_2)'$. In that case, the pair $(\mathtt{g}_1, \mathtt{g}_2)$ is dropped. This is why the pre-processing times for formal contexts and clarified formal contexts are the same in Table 4.13. The processing of the clarified formal context is the more efficient by far given the great reduction of objects it performs in the formal context. For example, for the *contraceptive* dataset, it reduces the number of pairs from 1 million to 1 thousand. A similar change can be then observed for the execution time of the concept lattice construction. The density of the clarified formal contexts is similar through all datasets (around 50%). This shows that the differences in calculation time are not due to the amount of information present in each formal context. This is further discussed at the end of this section.

An interesting exception occurs with dataset *pglw00*. It contains 17995 (10^4) objects meaning that the creation of the formal context should contain more than 10^8 objects (actually, the figure is over 161 millions). This sheer size of elements makes prohibitive the calculation of the formal context and the clarified formal context. For example, consider that each object can be represented by only a single integer variable with size 8 bytes, then the whole context would have a size of around 12 GB of memory (best case). This is particularly interesting considering that the formal context contains only 6 attributes, meaning that the concept lattice can only contain up to 64 formal concepts (which it does). Through the use of pattern structures we can obtain the formal concepts of *pglw00* in less than a minute and the size of the concept lattice in a compact notation is 13 MB of memory. This makes clear that, while the use of binarization and clarification of the dataset is very useful for small datasets, for slightly larger ones (consider that *pglw00* it is only 1 order of magnitude larger than *caulkins* or *contraceptive*), this technique is no longer feasible.

Table 4.14 gives a few statistics about pattern intents of the pattern concept lattices. For each

[64] `http://openmp.org/`

dataset it shows the average (and standard deviation) of the number of elements per component, as well as the average of the number of tolerance blocks. For the dataset *caulkins* we can see that, in average, a pattern intent contains 551 components. This mean that, in average, we should make more than 300K set intersection computations to obtain a single closure. This explains the fact that, even with parallelization, the calculation of the *caulkins* dataset concept lattice using partition pattern structures takes over 5 hours. On the other side, *pgwl00*, while containing 10 times more objects, has an average of 88 components per intent meaning around 8K intersections per closure computation. This is due to the fact that *pgwl00* only has one numerical attribute and half the total number of attributes than *caulkins*. If we compare *caulkins* with *hughes-r* having both the same number of attributes, we can see how the processing of the *hughes-r* dataset requires a fraction of the time for processing *caulkins*. It is worth noticing that even when *hughes-r* has a quarter of the number of *caulkins*'s objects (401 and 1685, respectively), this does not explain the difference in computation time and number of formal concepts. In fact, the clarified formal contexts for both datasets have similar sizes (1146×12 for *caulkins* and 1054×12 for *hughes-r*). Furthermore, it cannot even be explained in terms of the density of the formal context for which we have a small difference as shown in Table 4.13.

The only factor which explain this difference is established in the number of numerical attributes, which for *caulkins* are 9 out of 12 or three times those of *hughes-r* with 3 out of 12 numerical attributes. In general, a numerical attribute with a given θ-value generates a partition with more components and fewer elements per component than a categorical attribute, which yields a partition with less large components. More components per pattern intent increment quadratically the number of intersections required per closure computation. This is confirmed by the statistics provided in Table 4.14 which shows that pattern intents in *hughes-r* have in average less large components than *caulkins*. This phenomenon is not easily graspable from the formal contexts nor the clarified version, however it can be understood as follows. For a large enough θ-value for a given attribute $\mathtt{m} \in \mathtt{M}$, every pair of objects will be similar for that attribute and thus, every pair of objects will have that attribute in the formal context. Consequently, the attribute $\mathtt{m}$ will subsume any other attribute in $\mathtt{M}$, meaning that the closure of $\mathtt{m}$ with any other attribute is the top concept of the lattice. Differently, for a small enough θ-value, most pairs of objects will be not similar w.r.t. a given attribute. Actually, if no pairs of attributes are similar for $\mathtt{m}$, the meet between $\gamma(\mathtt{m})$ and $\gamma(\mathtt{n})$ for any $\mathtt{n} \in \mathtt{M}$, will be the bottom concept of the lattice (if we do not consider pairs $(\mathtt{g}_i, \mathtt{g}_j)$ with $\mathtt{g}_i = \mathtt{g}_j$). Hence, it is for middle values of θ that we have the maximum number of possible concepts in the lattice. It is worth noticing that, while this is also true for categorical attributes, i.e. a unique category is equal to a large enough θ-value and a single category per object is similar to a low enough θ-value, the difference remains in the sizes of the search spaces. Categorical attributes yield partitions, while numerical attributes yield tolerance blocks (with overlaps). The later, has a far larger search space than the first.

Finally, under the evidence shown by the experimental results, we can conclude that the use of pattern structures is of critical importance for mining similarity dependencies in medium-large datasets, where binarization and clarification are not possible due to computational limitations. Nevertheless, for sufficiently small datasets, the evidence shows that using standard FCA is a far better option. We have also shown how setting the values of θ can greatly influence the output of our algorithm. The scripts and binaries necessary to replicate the experiments are freely available online[65].

[65] `http://liris.cnrs.fr/mehdi.kaytoue/alg/ijgs2014.experiments.tbz`

Dataset	θ-values
Diagnosis	$\theta_1 = 0.3, \theta_{2,3,4,5,6,7,8} = 0$
Contraceptive	$\theta_1 = 5, \theta_{2,3,4,5,6,7,8,9} = 0$
Servo	$\theta_{1,2,4,5} = 0, \theta_3 = 5$
Caulkins	$\theta_1 = 5, \theta_{2,3,4} = 0, \theta_{5,12} = 300, \theta_{6,7,10} = 2000, \theta_{8,9,11} = 10$
Hughes-r	$\theta_{1,2,3}, \theta_{4,5,6,7,8,9,10,11,12} = 0$
Pglw00	$\theta_1 = 5, \theta_{2,3,4,5,6} = 0$

Table 4.12: Theta values for each dataset. The subindex of the θ-value corresponds to the attribute it was applied on. Comma separated values indicate more than one attribute.

Pattern structures $(M, (D, \sqcap), \delta)$							
Dataset	$\lvert M\rvert$	$\lvert T\rvert$=$\lvert G\rvert$	#Con.	Num.	Cat.	Exec. Time [s]	
Diagnosis	8	120	98	1	7	0.81	
Contraceptive	10	1473	1024	1	9	734.2	
Servo	5	167	28	1	4	1.30	
Caulkins	12	1685	2704	9	3	19783.3	
Hughes-r	12	401	754	3	9	24.35	
Pglw00	6	17995	64	1	5	55.84	
Formal context $(\mathfrak{B}_2(G), M, I)$							
Dataset	$\lvert G\rvert$	$\lvert M\rvert$	#Con.	Num.	Cat.	Den. [%]	Exec. Time [s]
Diagnosis	7082	8	98	1	7	48.5	0.32 + 0.09
Contraceptive	1082307	10	1024	1	9	44.8	120.46 + 47.6
Servo	13688	5	28	1	4	38.1	0.54 + 0.1
Caulkins	1412827	12	2704	9	3	43.4	168.19 + 102.249
Hughes-r	80200	12	754	3	9	50.4	5.11 + 3.85
Pglw00	161892017	6	64	1	5	-	-
Formal context $(\mathfrak{B}_2(G), M, I)$ **clarified**							
Dataset	$\lvert G\rvert$	$\lvert M\rvert$	#Con.	Num.	Cat.	Den. [%]	Exec. Time [s]
Diagnosis	50	8	98	1	7	48.5	0.32 + 0.02
Contraceptive	1017	10	1024	1	9	50.0	120.46 + 0.089
Servo	25	5	28	1	4	48.4	0.54 + 0.006
Caulkins	1146	12	2704	9	3	47.4	168.19 + 0.169
Hughes-r	1054	12	754	3	9	49.5	5.11 + 0.063
Pglw00	-	6	64	1	5	-	-

Table 4.13: Datasets and execution times (Con. : Concepts. Num. : Numerical attributes. Cat. : Categorical attributes. Den. : Density ($\frac{|I|}{|G|\times|M|}$)). The symbol "-" indicates that the value could not be obtained by computational limitations.

Pattern structures				
Dataset	Mean elements	STD elements	Mean components	STD components
Diagnosis	19.63	15.72	25.72	25.01
Contraceptive	24.09	60.51	416.77	298.65
Servo	20.12	24.43	33.5	40.19
Caulkins	12.69	43.39	551.66	173.99
Hughes-r	29.36	31.66	25.46	20.49
Pglw00	1616.13	2008.25	88.3	139.16

Table 4.14: Statistics over the tolerance blocks in the pattern intents

4.6 Conclusions

In this chapter we have described a formal framework for biclustering using partition pattern structures. This model is inspired by the biclustering-like process derived from heterogeneous pattern structures.

We have described different analogies to understand biclusters. Let us recall them here.

- Biclusters capture the dual relation between objects and attributes. Its main difference with FCA is that it involves values.
- Biclusters capture the coherent variation of objects under a subset of circumstances.
- Biclusters capture numerical patterns in a "local scale" rather than the "global scale" captured by standard clustering.
- Biclusters are a submatrices the values of which follow a given restriction.

In our model we have described biclusters in terms of features of a concept lattice of partition pattern concepts. We have gone as far as to extend this model to work with tolerance blocks, while stating that this is a natural generalization derived from the connections between tolerance and equivalence relations. Moreover, we have generalized our approach to interval pattern structures and triadic concept analysis, exposing the deep relation between biclustering as a data mining technique and the formal concept analysis framework.

By exploiting this relation we are able to use the machinery of FCA to perform more advanced tasks for the benefit of biclustering. We have made this clear by showing two applications of our biclustering model, namely recommender systems and functional dependency mining. We have shown how, using heterogeneous descriptions which allow combining biclusters and tag annotations, we are able to contextualize recommendations and integrate different sources of information. In the case of functional dependencies, we have shown how the partition pattern structure paradigm can used to mine similarity dependencies (i.e. functional dependencies based on a tolerance instead of an equivalence relation) and how, notions as attribute implications and concept stability, can be used to aid the navigation of the search space.

Conclusions and perspectives

Summary

Information retrieval is almost as old as informatics itself, and it is probably one of the firsts real applications of information theory for the classification of large document collections. On the other hand, formal concept analysis, a mathematical framework for data mining and knowledge discovery, was in certain way one of the first models supporting information retrieval to compute answers for user queries. Even now, we understand FCA as a natural implementation of the Boolean model for retrieval.

Both techniques have been linked from long ago and currently, even though this bond is weaker than it used to be, their intersection continues to be an interesting field of research with applications in a myriad of domains spanning multimedia indexing, linked open data and file systems.

In this work we have presented our contributions to IR using FCA techniques focusing them in its two main sub-processes, namely "retrieval" and "indexing". Our contributions on retrieval are based on a clever process design for using the concept lattice as an index of documents. This process, named CLAIRE (after Concept Lattices for Information REtrieval), is framed in the knowledge discovery in databases process description, since document retrieval can be considered as a knowledge discovery task itself. To fully exploit the internal concept lattice structure, we have proposed a novel navigation-for-retrieval strategy based on cousin concepts. This strategy allows modifying a given user query using the notions of case-based reasoning and semantic similarity between formal concepts. Experimental evidence suggests that this approach is better than standard retrieval techniques and state-of-the-art FCA-based IR systems.

As the IR community moved forward the Boolean retrieval model, FCA became less relevant as an alternative for building retrieval systems, mainly due to the fact that current indexes are built from complex document descriptions, such as numerical representations. Our contributions on indexing refer initially to the adaptation of CLAIRE to support the dynamics of advanced retrieval systems such as the vector space model. We achieved this by implementing the pattern structure framework, an extension of FCA that deals with complex object descriptions. We have shown how the concept lattice can derive a natural ranking in the documents retrieved for a given user, in the same way that the vector space model does. This extension is called ip-CLAIRE (after "interval pattern-CLAIRE"). While ip-CLAIRE is able to support more advanced IR techniques, it also loses what made FCA an interesting alternative for IR systems, mainly the ability to enrich document descriptions using external knowledge sources, and the ability to provide relevance feedback guiding the user search experience. In a second contribution on indexing, we have presented the heterogeneous pattern structures framework, as a mean to index documents simultaneously in different and distinct spaces, i.e. spaces that are orthogonal and different in nature, e.g. numerical, sets, intervals, etc. Through the use of heterogeneous pattern structures we have been able to create complex indices where documents can be described by

convex regions in an arbitrary vector space as well as elements inside a taxonomy.

Our final contributions go beyond indexing in the sense that they are also applicable for data mining. In a certain way, it is difficult to differentiate between indexing and data mining techniques, without knowing the main goal of the application they support. For this reason, we have decided to apply our experience in the field of FCA-based IR systems for data mining purposes. To achieve this, we have proposed a biclustering algorithm based on the lessons learnt from the heterogeneous pattern structures model and particularly, in the definitions given for partition pattern structures. Biclustering have been widely used for gene expression mining, information retrieval and more recently, for functional dependency mining. We have shown that biclusters can be characterized through standard FCA techniques, going as far as to introduce three different models, namely biclustering based on partition pattern structures, interval pattern structures and triadic concept analysis. Finally, we have presented two applications for the biclustering model proposed: a proposition for recommender systems and a generalization of the model for functional dependency mining and similar dependencies.

Perspectives

In Chapter 2 we have presented CLAIRE as a model for IR using FCA, case-based reasoning and semantic enrichment. An interesting perspective of this work is being able to support more complex semantic applications such as ontologies and description logics (DL) [134, 5]. Arguably, the way in that queries are extended through the use of cousin concepts and semantic similarity can be considered as a form of *inferencing* based on an external knowledge source (in our case, a dictionary). A good question is if this procedure can be replaced by an actual ontology-based inferencing. Information supported over ontologies and the resource description framework (RDF) is called linked open data [13] and its retrieval is an active topic of research [3, 58].

In the case of ip-CLAIRE and the heterogeneous pattern structures model, an interesting research perspective would be to study their possible extension to support different IR dynamics. As we have described, ip-CLAIRE was proposed to support the vector space model of retrieval in which documents and queries are compared through distances in the description space. Differently, in the probabilistic model of retrieval, documents and queries are evaluated by probability functions [93] which allow predicting "how likely is this document to be relevant for that query". In ip-CLAIRE, we used the notion of upper-bounding the distance of documents within a formal concept w.r.t. the query. This notion can be mapped to the probabilistic model by upper-bounding the probability of a document within a formal concept to be relevant for the query. This idea seems conceptually similar to *concept stability* which has been used to measure the probability of a formal concept to maintain its intent if arbitrary objects are removed from its extent [125]. Furthermore, approximations of stability bounded to intervals have been studied in [17].

Two model extensions for the biclustering models presented are of high interest in the research community. Firstly, the generalization of our model to mine biclusters with more than two dimensions, or what has been called "high-order co-clustering" [30] (consider that in three-dimensional spaces we can mine co-clusters in the shape of cubes). Cubes of data are a central element in data analysis for business intelligence, e.g. OLAP cubes [32]. Secondly, the combination of heterogeneous pattern structures with our biclustering model can be used to support heterogeneous bicluster mining. We can think of a heterogeneous bicluster as one where its attributes have a different nature, i.e. numerical and categorical data. This kind of patterns is frequent in linked open data, where objects can be described by a variety of data types [60, 110].

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *SIGMOD Record*, volume 22, pages 207–216, New York, NY, USA, 1993. ACM.

[2] R. AL-Msie'deen, A. D. Seriai, M. Huchard, C. Urtado, and S. Vauttier. Mining features from the object-oriented source code of software variants by combining lexical and structural similarity. In C. Zhang, J. Joshi, E. Bertino, and B. Thuraisingham, editors, *Proceedings of the 14th IEEE International Conference on Information Reuse and Integration (IRI 2013)*, pages 586–593, San Francisco, USA, 2013. IEEE.

[3] M. Alam and A. Napoli. Defining Views with Formal Concept Analysis for Understanding SPARQL Query Results. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications*, pages 255–266, 2014.

[4] F. Alqadah, C. Reddy, J. Hu, and H. Alqadah. Biclustering neighborhood-based collaborative filtering method for Top-n recommender systems. *Knowledge and Information Systems*, 22:475–491, 2014.

[5] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[6] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[7] J. Baixeries, M. Kaytoue, and A. Napoli. Computing functional dependencies with pattern structures. In L. Szathmary and U. Priss, editors, *CLA*, volume 972 of *CEUR Workshop Proceedings*, pages 175–186. CEUR-WS.org, 2012.

[8] J. Baixeries, M. Kaytoue, and A. Napoli. Computing similarity dependencies with pattern structures. In M. Ojeda-Aciego and J. Outrata, editors, *CLA*, volume 1062 of *CEUR Workshop Proceedings*, pages 33–44. CEUR-WS.org, 2013.

[9] J. Baixeries, M. Kaytoue, and A. Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of Mathematics and Artificial Intelligence*, pages 1–21, 2014.

[10] M. Barbut and B. Monjardet. *Ordre et classification : Algèbre et combinatoire*. 1970.

[11] R. Bendaoud, Y. Toussaint, and A. Napoli. Pactole: A methodology and a system for semi-automatically enriching an ontology from a collection of texts. In *Proceedings of*

the 16th international conference on Conceptual Structures: Knowledge Visualization and Reasoning, pages 203–216, 2008.

[12] J. Besson, C. Robardet, L. Raedt, and J.-F. Boulicaut. Mining Bi-sets in Numerical Data. In S. Džeroski and J. Struyf, editors, *Knowledge Discovery in Inductive Databases*, volume 4747 of *Lecture Notes in Computer Science*, pages 11–23. Springer Berlin Heidelberg, 2007.

[13] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[15] F. J. M. Bosman, R. W. T. Bouwman, and P. D. Bruza. The Effectiveness of Navigable Information Disclosure Systems. In *Proceedings of the Informatiewetenschap*. University of Nijmegen, Dept. of Informatics, 1991.

[16] R. J. Brachman and T. Anand. The Process of Knowledge Discovery in Databases. In *Advances in Knowledge Discovery and Data Mining*, pages 37–57. 1996.

[17] A. Buzmakov, S. O. Kuznetsov, and A. Napoli. Scalable estimates of concept stability. In C. V. Glodeanu, M. Kaytoue, and C. Sacarea, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8478 LNAI of *Lecture Notes in Computer Science*, pages 157–172. Springer Berlin Heidelberg, 2014.

[18] C. Carpineto, S. Mizzaro, G. Romano, and M. Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology*, 60(5):877–895, 2009.

[19] C. Carpineto and G. Romano. Galois : An order-theoretic approach to conceptual clustering. *Proceedings of the 10th International Conference on Machine Learning (ICML'93)*, pages 33–40, 1993.

[20] C. Carpineto and G. Romano. ULYSSES: A lattice-based multiple interaction strategy retrieval interface. In B. Blumenthal, J. Gornostaev, and C. Unger, editors, *Human-Computer Interaction*, volume 1015 of *Lecture Notes in Computer Science*, pages 91–104. Springer Berlin Heidelberg, 1995.

[21] C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122, Aug. 1996.

[22] C. Carpineto and G. Romano. Information retrieval through hybrid navigation of lattice representations. *International Journal of Human-Computer Studies*, 45(5):553 – 578, 1996.

[23] C. Carpineto and G. Romano. Effective reformulation of Boolean queries with concept lattices. In T. Andreasen, H. Christiansen, and H. Larsen, editors, *Flexible Query Answering Systems*, volume 1495 of *Lecture Notes in Computer Science*, pages 83–94. Springer Berlin Heidelberg, 1998.

[24] C. Carpineto and G. Romano. Order theoretical ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.

[25] C. Carpineto and G. Romano. Exploiting the potential of concept lattices for information retrieval with CREDO. *Journal of Universal Computer Science*, 10:985 – 1013, 2004.

[26] C. Carpineto and G. Romano. Using concept lattices for text retrieval and mining. *Formal Concept Analysis*, pages 161–179, 2005.

[27] A. Castellanos, A. García-Serrano, and J. Cigarrán. Linked Data-based Conceptual Modelling for Recommendation: A FCA-Based Approach. In M. Hepp and Y. Hoffner, editors, *E-Commerce and Web Technologies*, volume 188 of *Lecture Notes in Business Information Processing*, pages 71–76. Springer International Publishing, 2014.

[28] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *ACM Trans. Knowl. Discov. Data*, 3(1):3:1–3:36, Mar. 2009.

[29] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 2000.

[30] A. Chiaravalloti, G. Greco, A. Guzzo, and L. Pontieri. An Information-Theoretic Framework for High-Order Co-clustering of Heterogeneous Objects. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 598–605. Springer Berlin Heidelberg, 2006.

[31] J. M. Cigarrán, J. Gonzalo, A. Peñas, and F. Verdejo. Browsing search results via formal concept analysis: Automatic selection of attributes. In P. Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 74–87. Springer Berlin Heidelberg, 2004.

[32] E. Codd, S. Codd, and C. Salley. *Providing OLAP (On-line Analytical Processing) to User-analysts: An IT Mandate*. Codd & Associates, 1993.

[33] V. Codocedo, C. Lopez, and H. Astudillo. Jump-starting a body-of-knowledge with a semantic wiki on a discipline ontology. In *5th Workshop on Semantic Wikis - Linking Data and People (SemWiki 2010) at the 7th Extended Semantic Web Conference (ESWC 2010), Hersonissos, Greece, May 31st, 2010. Proceedings.*, volume 632 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[34] V. Codocedo, I. Lykourentzou, H. Astudillo, and A. Napoli. Using pattern structures to support information retrieval with formal concept analysis. In *Proceedings of the International Workshop "What can FCA do for Artificial Intelligence?" (FCA4AI at IJCAI 2013), Beijing, China, August 5, 2013.*, pages 15–24, 2013.

[35] V. Codocedo, I. Lykourentzou, and A. Napoli. A Contribution to Semantic Indexing and Retrieval Based on FCA - An Application to Song Datasets. In *Proceedings of The Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11-14, 2012*, pages 257–268, 2012.

[36] V. Codocedo, I. Lykourentzou, and A. Napoli. A semantic approach to concept lattice-based information retrieval. *Annals of Mathematics and Artificial Intelligence*, pages 1–27, 2014.

[37] V. Codocedo and A. Napoli. A Proposition for Combining Pattern Structures and Relational Concept Analysis. In C. Glodeanu, M. Kaytoue, and C. Sacarea, editors, *Formal Concept Analysis*, volume 8478 of *Lecture Notes in Computer Science*, pages 96–111. Springer International Publishing, 2014.

[38] V. Codocedo and A. Napoli. Lattice-based biclustering using partition pattern structures. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 213–218, 2014.

[39] V. Codocedo and A. Napoli. Formal Concept Analysis and Information Retrieval - A Survey. In J. Baixeries, C. Sacarea, and M. Ojeda-Aciego, editors, *Proceedings of the 13th International Conference on Formal Concept Analysis*, volume 9113 of *Lecture Notes in Computer Science*, pages 61–77. Springer, 2015.

[40] R. Cole and P. Eklund. Application of Formal Concept Analysis to Information Retrieval using a Hierarchically Structured Thesaurus. In *International Conference on Conceptual Graphs, ICCS '96, University of New South*, pages 1–12, 1996.

[41] R. Cole and P. Eklund. Analyzing an Email Collection Using Formal Concept Analysis. In J. Żytkow and J. Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1704 of *Lecture Notes in Computer Science*, pages 309–315. Springer Berlin Heidelberg, 1999.

[42] R. J. Cole, P. W. Eklund, and G. Stumme. Document Retrieval for Email Search and Discovery using Formal Concept Analysis. *Journal of Applied Artificial Intelligence (AAI)*, 17(3):257–280, 2003.

[43] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):1097–4571, 1990.

[44] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[45] S. Dominich. *The Modern Algebra of Information Retrieval*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[46] D. Dubois, F. D. de Saint-Cyr, and H. Prade. A Possibility-Theoretic View of Formal Concept Analysis. *Fundamenta Informaticae*, 75(1-4):195–213, 2007.

[47] P. Duboucherryan and D. Bridge. Collaborative Recommending using Formal Concept Analysis. *Knowledge-Based Systems*, 19(5):309–315, Sept. 2006.

[48] J. Ducrou. Dvdsleuth: A case study in applied formal concept analysis for navigating web catalogs. In U. Priss, S. Polovina, and R. Hill, editors, *Conceptual Structures: Knowledge Architectures for Smart Applications*, volume 4604 of *Lecture Notes in Computer Science*, pages 496–500. Springer Berlin Heidelberg, 2007.

[49] J. Ducrou and P. Eklund. SearchSleuth: The Conceptual Neighbourhood of a Web Query. In *Proceedings of the 2007 International Conference on Concept Lattices and their Applications*, CLA '07, pages 253–263, 2007.

[50] J. Ducrou, B. Vormbrock, and P. Eklund. FCA-Based Browsing and Searching of a Collection of Images. In H. Schärfe, P. Hitzler, and P. Ø hrstrøm, editors, *Conceptual Structures: Inspiration and Application*, volume 4068 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2006.

[51] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*, pages 281–285, New York, New York, USA, May 1988. ACM Press.

[52] P. Eklund and J. Ducrou. Navigation and annotation with formal concept analysis. In D. Richards and B.-H. Kang, editors, *Knowledge Acquisition: Approaches, Algorithms and Applications*, volume 5465 of *Lecture Notes in Computer Science*, pages 118–121. Springer Berlin Heidelberg, 2009.

[53] P. Eklund, J. Ducrou, and P. Brawn. Concept Lattices for Information Visualization: Can Novices Read Line-Diagrams? In P. Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 57–73. Springer Berlin Heidelberg, 2004.

[54] R. A. Fairthorne. The patterns of retrieval. *American Documentation*, 7(2):65–70, 1956.

[55] S. Ferré and A. Hermann. Reconciling faceted search and query languages for the semantic web. *IJMSO*, 7(1):37–54, 2012.

[56] S. Ferré and O. Ridoux. A File System Based on Concept Analysis. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. Pereira, Y. Sagiv, and P. Stuckey, editors, *Computational Logic - CL 2000*, volume 1861 of *Lecture Notes in Computer Science*, pages 1033–1047. Springer Berlin Heidelberg, 2000.

[57] S. Ferré and O. Ridoux. A Logical Generalization of Formal Concept Analysis. In B. Ganter and G. W. Mineau, editors, *ICCS*, volume 1867 of *LNCS*, pages 357–370, 2000.

[58] S. Ferré and A. Hermann. Semantic search: Reconciling expressive querying and exploratory search. In *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 177–192. Springer Berlin Heidelberg, 2011.

[59] A. Formica. Concept similarity in Formal Concept Analysis: An information content approach. *Knowledge-Based Systems*, 21(1):80–87, Feb. 2008.

[60] T. Franz, A. Schultz, S. Sizov, and S. Staab. Triplerank: Ranking semantic web data by tensor decomposition. In A. Bernstein, D. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 213–228. Springer Berlin Heidelberg, 2009.

[61] A. V. Freitas, W. Ayadi, M. Elloumi, J. Oliveira, J. Oliveira, and J.-K. Hao. *Survey on Biclustering of Gene Expression Data.* 2013.

[62] B. Ganter and S. O. Kuznetsov. Pattern Structures and their projections. *Conceptual Structures: Broadening the Base*, 2001.

[63] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Dec. 1999.

[64] A. Ginsberg. A unified approach to automatic indexing and information retrieval. *IEEE Expert*, 8(5):46–56, Oct. 1993.

[65] R. Godin, J. Gecsei, and C. Pichet. Design of a Browsing Interface for Information Retrieval. In *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '89, pages 32–39, New York, NY, USA, 1989. ACM.

[66] R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38(5):747–767, May 1993.

[67] R. Godin, E. Saunders, and J. Gecsei. Lattice model of browsable data spaces. *Information Sciences: an International Journal*, 40(2):89–116, Dec. 1986.

[68] F. A. Grootjen and T. van der Weide. Conceptual relevance feedback. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 2, pages 471–476, Oct. 2002.

[69] F. A. Grootjen and T. P. van der Weide. Conceptual query expansion. *Data Knowl. Eng.*, 56(2):174–193, Feb. 2006.

[70] T. Hofmann. Probabilistic latent semantic indexing. pages 50–57, 1999.

[71] M. Huchard, M. R. Hacene, C. Roume, and P. Valtchev. Relational concept discovery in structured datasets. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):39–76, June 2007.

[72] D. I. Ignatov, A. Y. Kaminskaya, N. Konstantinova, A. Malyukov, and J. Poelmans. FCA-Based Recommender Models and Data Analysis for Crowdsourcing Platform Witology. In *Graph-Based Representation and Reasoning - 21st International Conference on Conceptual Structures, {ICCS} 2014, Ia{\c{s}}i, Romania, July 27-30, 2014, Proceedings*, pages 287–292, 2014.

[73] D. Ignatov I. and S. O. Kuznetsov. Concept-based Recommendations for Internet Advertisement. In *Proceedings of the 2008 International Conference on Concept Lattices and their Applications*, 2008.

[74] P. Ingwersen. Cognitive perspectives of information retrieval interaction: Elements of a cognitive IR theory. *Journal of Documentation*, 52(1):3–50, 1996.

[75] M. Kaytoue, Z. Assaghir, A. Napoli, and S. O. Kuznetsov. Embedding tolerance relations in formal concept analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, pages 1689–1692, New York, New York, USA, Oct. 2010. ACM Press.

[76] M. Kaytoue, V. Codocedo, J. Baixeries, and A. Napoli. Three Interrelated {FCA} Methods for Mining Biclusters of Similar Values on Columns. In K. Bertet and S. Rudolph, editors, *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Ko{š}ice, Slovakia, October 7-10, 2014.*, volume 1252 of *{CEUR} Workshop Proceedings*, pages 243–254. CEUR-WS.org, 2014.

[77] M. Kaytoue, S. O. Kuznetsov, J. Macko, and A. Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, 2013.

[78] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Biclustering numerical data in formal concept analysis. In *Formal Concept Analysis*. 2011.

[79] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Revisiting numerical pattern mining with formal concept analysis. *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, pages 1342–1347, Nov. 2011.

[80] M. Kim and P. Compton. Formal concept analysis for domain-specific document retrieval systems. In M. Stumptner, D. Corbett, and M. Brooks, editors, *AI 2001: Advances in Artificial Intelligence*, pages 237–248. Springer Berlin Heidelberg, 2001.

[81] M. Kim and P. Compton. Evolutionary document management and retrieval for specialized domains on the web. *International Journal of Human-Computer Studies*, 60(2):201–241, 2004.

[82] B. Koester. Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies. In P. Perner, editor, *Industrial Conference on Data Mining*, volume 4065 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2006.

[83] S. O. Kuznetsov. Galois connections in data analysis: Contributions from the soviet era and modern russian research. In *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*. 2005.

[84] S. O. Kuznetsov. On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):101–115, 2007.

[85] S. O. Kuznetsov. Pattern Structures for Analyzing Complex Data. In *Proceedings of the 12th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 5908 of *Lecture Notes in Computer Science*, pages 33–44. Springer Berlin Heidelberg, Dec. 2009.

[86] S. O. Kuznetsov. Fitting Pattern Structures to Knowledge Discovery in Big Data. In P. Cellier, F. Distel, and B. Ganter, editors, *Formal Concept Analysis*, volume 7880 of *Lecture Notes in Computer Science*, pages 254–266. Springer Berlin Heidelberg, 2013.

[87] S. O. Kuznetsov and S. A. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.

[88] F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In G. Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *Conceptual Structures: Applications, Implementation and Theory*, volume 954 of *Lecture Notes in Computer Science*, pages 32–43. Springer Berlin Heidelberg, 1995.

[89] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[90] S. Lopes, J.-M. Petit, and L. Lakhal. Functional and approximate dependency mining: database and FCA points of view. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):93–114, 2002.

[91] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, Jan. 2004.

[92] H. Mannila and K.-J. Räihä. *The Design of Relational Databases*. 1992.

[93] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. July 2008.

[94] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[95] B. Martin. Formal concept analysis and semantic file systems. In P. Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 88–95. Springer Berlin Heidelberg, 2004.

[96] J. Martinez and E. Loisant. Browsing Image Databases with Galois' Lattices. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 791–795, New York, NY, USA, 2002. ACM.

[97] N. Messai, M.-D. Devignes, A. Napoli, and M. Smaïl-Tabbone. Querying a bioinformatic data sources registry with concept lattices. In *Proceedings of the 13th international conference on Conceptual Structures: common Semantics for Sharing Knowledge*, volume 3596 of *Lecture Notes in Computer Science*, July 2005.

[98] N. Messai, M.-D. Devignes, A. Napoli, and M. Smail-Tabbone. Many-Valued Concept Lattices for Conceptual Clustering and Information Retrieval. In *Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 127–131, June 2008.

[99] N. Messai, M.-D. Devignes, A. Napoli, and M. Smaïl-Tabbone. BR-Explorer: An FCA-based algorithm for Information Retrieval. In *Fourth International Conference On Concept Lattices and Their Applications - CLA 2006*, Hammamet/Tunisia, 2006.

[100] N. Messai, M.-D. Devignes, A. Napoli, and M. Smaïl-Tabbone. Using Domain Knowledge to Guide Lattice-based Complex Data Exploration. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 847–852, 2010.

[101] L. Miclet, N. Barbot, and H. Prade. From analogical proportions in lattices to proportional analogies in formal concepts. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 627–632, 2014.

[102] A. Mili, R. Mili, and R. T. Mittermeir. Storing and retrieving software components: A refinement based system. In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 91–100, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[103] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, Nov. 1995.

[104] S. Miyamoto. Lattice-valued hierarchical clustering for analyzing information systems. In *Rough Sets and Current Trends in Computing*, volume 4259 of *Lecture Notes in Computer Science*. 2006.

[105] C. N. Mooers. *A Mathematical Theory of Language Symbols in Retrieval*. Zator Company, 1958.

[106] A. Napoli, C. Laurenço, and R. Ducournau. An object-based representation system for organic synthesis planning. *International Journal of Human-Computer Studies*, 41(1/2):5–32, 1994.

[107] E. Nauer and Y. Toussaint. Dynamical modification of context fo an iterative and interactive information retrieval process on the Web. In *Proceedings of the 2007 International Conference on Concept Lattices and their Applications*, CLA '07, pages 217–228, 2007.

[108] E. Nauer and Y. Toussaint. CreChainDo: an iterative and interactive Web information retrieval system based on lattices. *International Journal of General Systems*, 38(4):363–378, 2009.

[109] T. T. Nguyen, S. C. Hui, and K. Chang. A lattice-based approach for mathematical search using Formal Concept Analysis. *Expert Systems with Applications*, 39(5):5820–5828, 2012.

[110] M. Nickel and V. Tresp. An analysis of tensor models for learning on structured data. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*, pages 272–287. Springer Berlin Heidelberg, 2013.

[111] D. W. Oard and A. R. Diekema. Cross-language information retrieval. *Annual review of information science and technology*, 33:223–256, 1998.

[112] G. Pandey, G. Atluri, M. Steinbach, C. L. Myers, and V. Kumar. An association analysis approach to biclustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[113] G. S. Pedersen. A Browser for Bibliographic Information Retrieval, Based on an Application of Lattice Theory. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 270–279, New York, NY, USA, 1993. ACM.

[114] G. S. Pedersen. Relationship lattices for information modelling. *Information Modelling and Knowledge Bases*, pages 155–173, 1994.

[115] G. Pio, M. Ceci, C. Loglisci, D. D'Elia, and D. Malerba. Hierarchical and Overlapping Co-Clustering of mRNA: miRNA Interactions. In *ECAI*, Frontiers in Artificial Intelligence and Applications, 2012.

[116] G. Pio, M. Ceci, C. Loglisci, D. D'Elia, and D. Malerba. A novel biclustering algorithm for the discovery of meaningful biological correlations between mirnas and mrnas. *EMBnet.journal*, 18(A), 2012.

[117] J. Poelmans, P. Elzinga, S. Viaene, and G. Dedene. Formal concept analysis in knowledge discovery: a survey. In *Proceedings of the 18th international conference on Conceptual*

structures: from information to intelligence, ICCS'10, pages 139–153, Berlin, Heidelberg, 2010. Springer-Verlag.

[118] J. Poelmans, D. I. Ignatov, S. Viaene, G. Dedene, and S. O. Kuznetsov. Text mining scientific papers: a survey on FCA-Based information retrieval research. In P. Perner, editor, *Proceedings of the 12th Industrial conference on Advances in Data Mining: applications and theoretical aspects*, volume 7377 of *Lecture Notes in Computer Science*, pages 273–287, Berlin, Heidelberg, July 2012. Springer Berlin Heidelberg.

[119] D. Poshyvanyk and A. Marcus. Combining Formal Concept Analysis with Information Retrieval for Concept Location in Source Code. In *15th IEEE International Conference on Program Comprehension (ICPC '07)*, pages 37–48. IEEE, June 2007.

[120] U. Priss. A graphical interface for document retrieval based on formal concept analysis. In *Proceedings of the 8th Midwest Artificial Intelligence and Cognitive Science Conference*, pages 66–70, 1997.

[121] U. Priss. Lattice-based Information Retrieval. *Knowledge Organization*, 27:132 – 142, 2000.

[122] U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology*, 40(1):521–543, Sept. 2007.

[123] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[124] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[125] C. Roth, S. Obiedkov, and D. Kourie. Towards concise representation for taxonomies of epistemic communities. *Concept Lattices and Their Applications*, 2008.

[126] M. Rouane-Hacene, M. Huchard, A. Napoli, and P. Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *Proceedings of ICFCA 2007*, pages 51–65. LNAI 4390, Springer, 2007.

[127] M. Rouane-Hacene, M. Huchard, A. Napoli, and P. Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81–108, Mar. 2013.

[128] G. Salton, E. A. Fox, and H. Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, Nov. 1983.

[129] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, 1975.

[130] S. Senatore and G. Pasi. Lattice Navigation for Collaborative Filtering by Means of (Fuzzy) Formal Concept Analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 920–926, New York, NY, USA, 2013. ACM.

[131] A. Shah and L. Caves. ConceptOntoFs: A Semantic File System for Interno. In *First International Workshop on Plan 9 and Inferno*. 2006.

[132] D. A. Simovici and C. Djeraba. *Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[133] A. Srivastava and M. Sahami. *Text Mining: Classification, Clustering, and Applications*. Chapman & Hall/CRC, 1st edition, 2009.

[134] S. Staab and R. Studer. *Handbook on Ontologies*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[135] P. Treeratpituk and J. Callan. Automatically Labeling Hierarchical Clusters. In *Proceedings of the 2006 International Conference on Digital Government Research*, dg.o '06, pages 167–176. Digital Government Society of North America, 2006.

[136] L. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[137] J. Ullman. *Principles of Database Systems and Knowledge-Based Systems, volumes 1–2*. 1989.

[138] F. J. Valverde and C. Pelaez-Moreno. System vs. Methods: an Analysis of the Affordances of Formal Concept Analysis for Information Retrieval. In *Proceedings of the Workshop Formal Concept Analysis Meets Information Retrieval (FCAIR 2013)*, 2013.

[139] D. van der Merwe, S. Obiedkov, and D. Kourie. AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In P. Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 205–206. Springer, Berlin/Heidelberg, 2004.

[140] R. Wille. Restructuring Lattice Theory: An approach based on hierarchies of concepts. In S. Ferré and S. Rudolph, editors, *Formal Concept Analysis*, volume 5548 of *Lecture Notes in Computer Science*, pages 314–339. Springer Berlin Heidelberg, 2009.

[141] T. Wray and P. Eklund. Using formal concept analysis to create pathways through museum collections. In S. Kuznetsov, A. Napoli, and S. Rudolph, editors, *Proceedings of the 3rd International Workshop "What can FCA do for Artificial Intelligence"?* 2014.

Résumé

Un des premiers modèles d'indexation de documents qui utilise des termes comme descripteurs était une structure de treillis, cela une vingtaine d'années avant l'arrivée de l'analyse formelle de concepts (FCA pour "Formal Concept Analysis"), qui s'affirme maintenant comme un formalisme théorique important et solide pour l'analyse de données et la découverte de connaissances. Actuellement, la communauté en recherche d'information (RI) s'intéresse particulièrement à des techniques avancées pour la recherche des documents qui relèvent des probabilités et des statistiques. En parallèle, l'intérêt de la communauté FCA au développement de techniques qui font avancer l'état de l'art en RI tout en offrant des fonctionnalités sémantiques lui est toujours bien vivant.

Dans cette thèse, nous présentons un ensemble de contributions sur ce que nous avons appelé les systèmes FCA de recherche d'information ("FCA-based IR systems"). Nous avons divisé nos contributions en deux parties, à savoir l'extraction et l'indexation. Pour la récupération, nous proposons une nouvelle technique qui exploite les relations sémantiques entre les descripteurs dans un corpus de documents. Pour l'indexation, nous proposons un nouveau modèle qui permet de mettre en œuvre un modèle vectoriel d'indexation des documents s'appuyant sur un treillis de concepts (ou treillis de Galois). En outre, nous proposons un modèle perfectionné pour l'indexation hétérogène dans lequel nous combinons le modèle vectoriel et le modèle de recherche booléen.

Finalement, nous présentons une technique de fouille de données inspiré de l'indexation des documents, à savoir un modèle d'énumération exhaustive des biclusters en utilisant la FCA. Le biclustering est une nouvelle technique d'analyse de données dans laquelle les objets sont liés via la similitude dans certains attributs de l'espace de description, et non pas par tous les attributs comme dans le "clustering" standard. En traduisant ce problème en termes d'analyse formelle de concepts, nous pouvons exploiter l'algorithmique associée à la FCA pour développer une technique d'extraction de biclusters de valeurs similaires. Nous montrons le très bon comportement de notre technique, qui fonctionne mieux que les techniques actuelles de biclustering avec énumération exhaustive.

Mots-clés: analyse formelle de concepts, recherche d'information, biclustering, relevance feedback, systèmes de recommandation

Abstract

One of the first models ever to be considered as an index for documents using terms as descriptors, was a lattice structure, a couple of decades before the arrival of Formal Concept Analysis (FCA) as a solid theory for data mining and knowledge discovery. While the Information Retrieval (IR) community has shifted to more advanced techniques for document retrieval, like probabilistic and statistic paradigms, the interest of the FCA community on developing techniques that would improve the state-of-the-art in IR while providing relevance feedback and semantic based features, never decayed.

In this thesis we present a set of contributions on what we call FCA-based IR systems. We have divided our contributions in two sets, namely retrieval and indexing. For retrieval, we propose a novel technique that exploits semantic relations among descriptors in a document corpus and a new concept lattice navigation strategy (called cousin concepts), enabling us to support classification-based reasoning to provide better results compared with state-of-the-art retrieval techniques. The basic notion in our strategy is supporting query modification using "term replacements" using the lattice structure and semantic similarity.

For indexing, we propose a new model that allows supporting the vector space model of retrieval using concept lattices. One of the main limitations of current FCA-based IR systems is related to the binary nature of the input data required for FCA to generate a concept lattice. We propose the use of pattern structures, an extension of FCA to deal with complex object descriptions, in order to support more advanced retrieval paradigms like the vector space model. In addition, we propose an advanced model for heterogeneous indexing through which we can combine the vector space model and the Boolean retrieval model. The main advantage of this approach is the ability of supporting indexing of convex regions in an arbitrary vectorial space built from a document collection.

Finally, we move forward to a mining model associated with document indexing, namely exhaustive bicluster enumeration using FCA. Biclustering is an emerging data analysis technique in which objects are related by similarity under certain attributes of the description space, instead of the whole description space like in standard clustering. By translating this problem to the framework of FCA, we are able to exploit the robust machinery associated with the computation of concept lattices to provide an algorithm for mining biclusters based on similar values. We show how our technique performs better than current exhaustive enumeration biclustering techniques.

Keywords: formal concept analysis, information retrieval, biclustering, relevance feedback, recommender systems